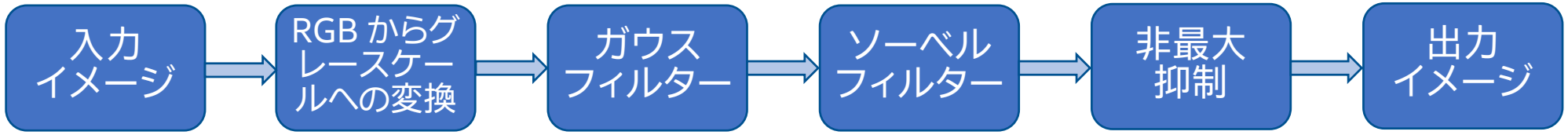
インテル® VTune™ Amplifier + OpenMP* によりスレッドのパフォーマンスとスケーラビリティーを向上する

Anoop Madhusoodhanan Prabha & Kenneth Craft

内容

- チューニングの可能性を調査する
  - 計算依存のワークロード (ソーベルフィルターと非最大抑制)
    - インテル® VTune™ Amplifier とインテル® Advisor – ベクトル化アドバイザーの使用
    - インテル® コンパイラーの最適化レポート
    - OpenMP* の SIMD プラグマを利用する外側のループのベクトル化
    - その他のコンパイラーによる最適化
    - OpenMP* によるスレッド化

エッジ検出アプリケーションのパイプライン



システム構成:

インテル® Core™ i7-7567U プロセッサー @ 3.50GHz
32GB RAM
インテル® C++ コンパイラー 17.0.2.174
GCC 7.1

インテル® VTune™ Amplifier の hotspot 解析レポート

Advanced Hotspots　Hotspots viewpoint (change)

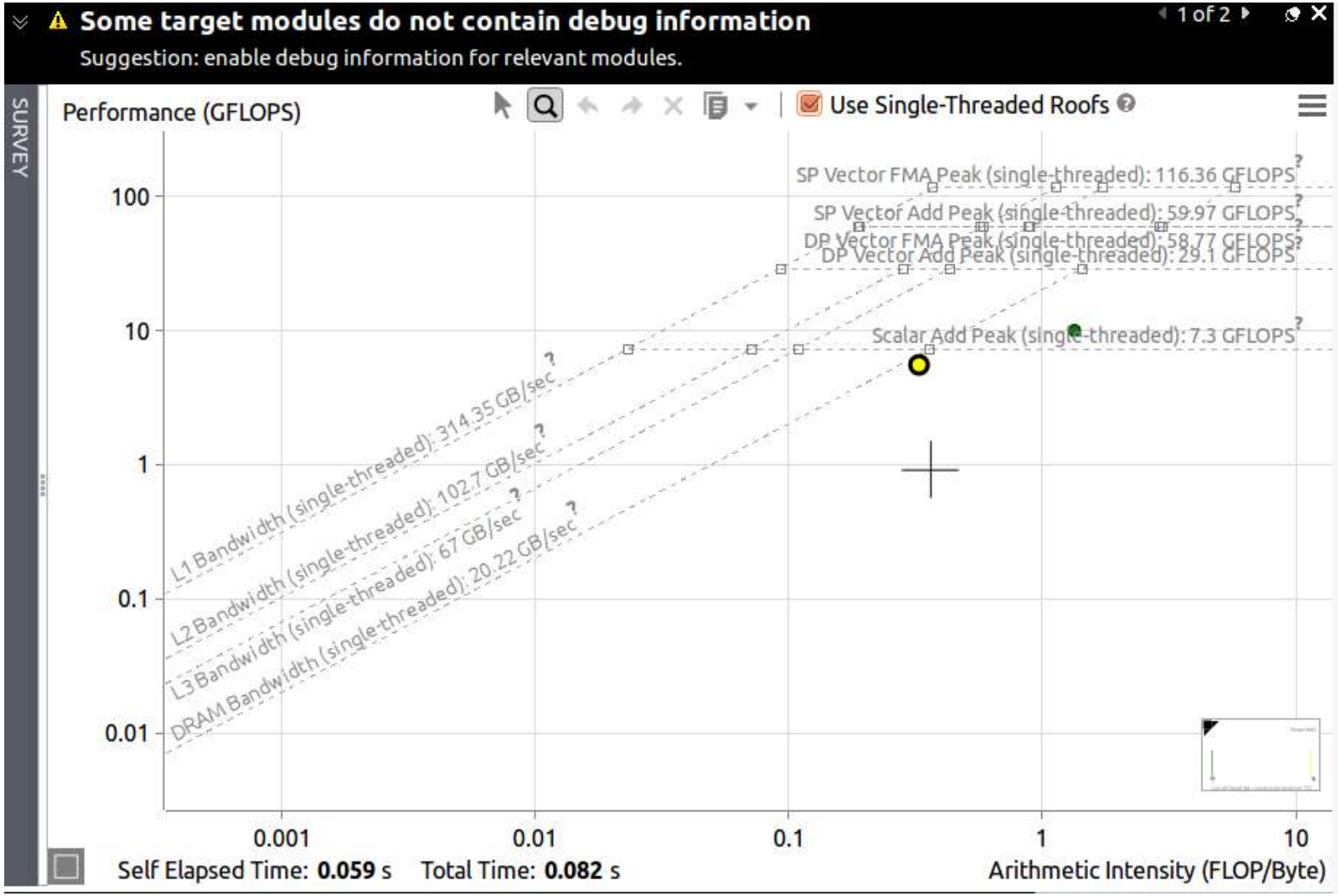
Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time ▼ Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	Spin Time	Overhead Time	Instructions Retired	CPI Rate
computeGradient	219.649ms	0ms	0ms	1,073,100,000	0.809
main	63.428ms	0ms	0ms	323,750,000	0.751
__libm_atan2f_l9	54.737ms	0ms	0ms	64,050,000	3.377
nonMaximumSupression	42.152ms	0ms	0ms	117,600,000	1.390
std::pow<float, int>	18.279ms	0ms	0ms	208,250,000	0.355
GaussianFilter	16.381ms	0ms	0ms	92,750,000	0.770
__libm_floor_y8	14.084ms	0ms	0ms	8,750,000	6.600
std::atan2	4.695ms	0ms	0ms	3,150,000	6.222
std::pow<float, int>	3.396ms	0ms	0ms	26,250,000	0.373
convertRGBtoGray	3.396ms	0ms	0ms	41,650,000	0.378
floor	1.898ms	0ms	0ms	350,000	28.000
atan2f	1.798ms	0ms	0ms	5,250,000	1.467
__libm_atan2f_e7	1.398ms	0ms	0ms	7,700,000	1.091
__intel_avx_rep_memset	1.099ms	0ms	0ms	0	
__libm_error_support	0.499ms	0ms	0ms	4,900,000	0.571
cv::Mat::~Mat	0.100ms	0ms	0ms	0	0.000

intel Software

インテル® Advisor のルーフライン解析



- ベクトル化アドバイザーのキャッシュを考慮したルーフライン解析は、ワークロードがメモリー依存または計算依存かを理解するのに役立つ
- 色付きの点はそれぞれワークロード中の関数を表す
- 点の位置から関数の特性が分かる
- X 軸は演算強度 (1 バイト・アクセスごとの FLOPS 数) を示し、この値が大きい場合ワークロードは計算依存であり、小さい場合はメモリー依存である
- ここでは黄色の点はソーベルフィルターを示し、この関数の FLOPS はスカラーのシングルスレッド・ピーク性能により制限される

ソーベルフィルターのコードとベースライン・パフォーマンス

```
for(int i = 1; i < rows-1; i++){
   int index = i*cols+1;
   int gindex = (i-1)*gcols;
   for(int j = 0; j < cols-2; j++){
      int index1 = index+j;
      int gindex1 = gindex+j;
      gradient_x = gradient_y = 0.0f;
      for(int k1 = -1; k1 <= 1; k1++){
         int index2 = index1+(k1*cols);
         for(int k2 = -1; k2 <=1; k2++){
            gradient_x += gradientx[k1+1][k2+1] * input.data[index2+k2];
            gradient_y += gradienty[k1+1][k2+1] * input.data[index2+k2];
         }
      }
      strength.data[gindex1] = sqrt(pow(gradient_x, 2) + pow(gradient_y, 2));
      degrees = floor(((atan2(gradient_y, gradient_x) * 180 / PI)/45) + 0.5f);
      direction.data[gindex1] = (degrees == 4)?0:(degrees*45);
   }
}
```

ベースライン・パフォーマンス

- 使用したコンパイラー・オプション: -O2 –qopt-report5 –qopt-report-phase=vec –std=c++11
- computeGradient の実行時間 = 0.33964 秒
- 非最大抑制の実行時間 = 0.0601606 秒

インテル® VTune™ Amplifier の全般解析レポート

General Exploration General Exploration viewpoint (change)

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Event Count | Platform

Grouping: Function / Call Stack

Function / Call Stack	Back-End Bound					
	Core Bound					
	Divider	Port Utilization				
		Cycles of 0 Ports Utilized	Cycles of 1 Port Utilized	Cycles of 2 Ports Utilized	Cycles of 3+ Ports Utilized	Vector Capacity Usage (FPU)
computeGradient	11.4%	26.3%	28.9%	13.9%	13.9%	13.7%
main	2.0%	10.1%	26.2%	20.2%	100.0%	12.5%
__libm_atan2f_l9	74.6%	0.0%	6.8%	24.9%	70.1%	25.0%
nonMaximumSupression	0.0%	2.7%	21.5%	32.2%	13.4%	0.0%

intel Software

インテル® Advisor による computegradient の調査解析結果

Summary | Survey & Roofline | Refinement Reports

Some target modules do not contain debug information

Suggestion: enable debug information for relevant modules.

1 of 2

ROOFLINE

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	Why No Vectorization?	Vectorized Loops: Vect...	Gain ...	VL.	Instruction Set Analysis: Traits	Data Types
computeGradient		0.051s	0.075s	Inlined Function						Float32
[loop in main at sample.cc:158]	1 Data type ...	0.013s	0.088s	Scalar Versions	1 outer loop was not a...				Divisions; Square Roo...	Float32; Float...
[loop in main at sample.cc:146]	1 Data type c...	0.005s	0.008s	Scalar Versions	1 outer loop was not au...				Type Conversions	Float32
GaussianFilter		0.003s	0.003s	Inlined Function						Float32
convertRGBtoGray		0.002s	0.002s	Inlined Function						
std::pow<float, int>		0.002s	0.002s	Inlined Function						

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

File: sample.cc:158 main

Line	Source	Total Time	%	Loop/Function Time	%	Traits
146	GaussianFilter(outputimage2, outputimage1);	8.000ms				
147	timer_stop1 = std::chrono::system_clock::now();					
148						
149	for(int i = 0; i < rows; i++)					
150	{					
151	int index1 = (i+1)*newcols+1;					
152	int index = i*cols;					
153	for(int j = 0; j < cols; j++)					
154	outputimage.data[index+j] = outputimage2.data[index1+j];	1.000ms				
155	}					
156	//Step 3 Gradient strength and direction					
157	timer_start2 = std::chrono::system_clock::now();					
158	computeGradient(outputimage2, gradientstrength, gradientdirection);	64.000ms		88.000ms		Divisions; Square Roots; Type Conversions

```
[loop in main at sample.cc:158]
  Scalar loop. Outer loop was not auto-vectorized: consider using SIMD directive
  No loop transformations applied
[loop in main at sample.cc:158]
  Scalar loop. Outer loop was not auto-vectorized: consider using SIMD directive
  Loop was distributed, chunk 2
```


intel Software

computeGradient に関するインテル® コンパイラーの最適化レポート

```
ループ開始 sample.cc(75,5)
    リマーク #15389: ベクトル化のサポート: 参照 gradientx[k1+1][k2+1] にアラインされていないアクセスが含まれています。[ sample.cc(77,20) ]
    リマーク #15389: ベクトル化のサポート: 参照 input->data[index2+k2] にアラインされていないアクセスが含まれています。[ sample.cc(77,44) ]
    リマーク #15389: ベクトル化のサポート: 参照 gradienty[k1+1][k2+1] にアラインされていないアクセスが含まれています。[ sample.cc(78,20) ]
    リマーク #15389: ベクトル化のサポート: 参照 input->data[index2+k2] にアラインされていないアクセスが含まれています。[ sample.cc(78,44) ]
    リマーク #15381: ベクトル化のサポート: ループ本体内でアラインされていないアクセスが使用されました。
    リマーク #15335: ループはベクトル化されませんでした: ベクトル化は可能ですが非効率です。オーバーライドするには vector always ディレクティブまたは
    -vec-threshold0 を使用してください。
    リマーク #15305: ベクトル化のサポート: ベクトル長 2
    リマーク #15309: ベクトル化のサポート: 正規化されたベクトル化のオーバーヘッド 2.273
    リマーク #15450: マスクなし非アライン･ユニット･ストライド･ロード: 4
    リマーク #15475: --- ベクトルのコストサマリー開始 ---
    リマーク #15476: スカラーのコスト: 19
    リマーク #15477: ベクトルのコスト: 11.000
    リマーク #15478: スピードアップの期待値: 0.620
    リマーク #15487: 型変換: 2
    リマーク #15488: --- ベクトルのコストサマリー終了 ---
ループの終了
```

インテル® Advisor の推奨事項

Summary | Survey & Roofline | Refinement Reports

Some target modules do not contain debug information

Suggestion: enable debug information for relevant modules.

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	Why No Vectorization?	Vectorized Loops: Vect...	Gain ...	VL	Instruction Set Analysis: Traits	Data Types
[loop in main at sample.cc:158]	1 Data type c...	0.013s	0.088s	Scalar	outer loop was not aut...				Divisions; Square Root...	Float32; Float64
[loop in main at sample.cc:146]	1 Data type c...	0.005s	0.008s	Scalar Versions	1 outer loop was not au...				Type Conversions	Float32

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

```
    integer, intent(in) :: n, n1
    real, intent(inout) :: a(n,n1)
    integer :: i, j
    do i=1,n
        do j=1,n
            a(j,i) = a(j-1,i)+1
        end do
    end do
end subroutine foo
```

Recommendations

- Run a Dependencies analysis to check if the loop has real dependencies. There are two types of dependencies:
  - True dependency - Read after write (RAW)
  - Anti-dependency - Write after read (WAR)
- If no dependencies exist, use one of the following to tell the compiler it is safe to vectorize:
  - Directive to prevent all dependencies in the loop

Target	ICL/ICC/ICPC Directive	IFORT Directive
Source Loop	#pragma simd or #pragma omp simd	!DIR$ SIMD or !$OMP SIMD

  - Directive to ignore only vector dependencies (which is safer)

Target	ICL/ICC/ICPC Directive	IFORT Directive
Source Loop	#pragma ivdep	!DIR$ IVDEP

  - `restrict` keyword
- If anti-dependency exists, use a directive where `k` is smaller than the distance between dependent items in anti-dependency. This enables vectorization, as dependent items are put into different ve

Target	ICL/ICC/ICPC Directive	IFORT Directive
Source Loop	#pragma simd vectorlength(k)	!DIR$ SIMD VECTORLENGTH(k)

- If using the `O3` compiler option, use a directive before the inner and outer loops to request vectorization of the outer loop:

Target	ICL/ICC/ICPC Directive	IFORT Directive
Inner loop	#pragma novector	!DIR$ NOVECTOR
Outer loop	#pragma vector always	!DIR$ VECTOR ALWAYS

intel Software

OpenMP* の simd プラグマによる明示的なベクトル化

```
for(int i = 1; i < rows-1; i++){
   int index = i*cols+1;
   int gindex = (i-1)*gcols;
   #pragma omp simd private(gradient_x, gradient_y, degrees)
   for(int j = 0; j < cols-2; j++){
      int index1 = index+j;
      int gindex1 = gindex+j;
      gradient_x = gradient_y = 0.0f;
      for(int k1 = -1; k1 <= 1; k1++){
         int index2 = index1+(k1*cols);
         for(int k2 = -1; k2 <=1; k2++){
            gradient_x += gradientx[k1+1][k2+1] * input.data[index2+k2];
            gradient_y += gradienty[k1+1][k2+1] * input.data[index2+k2];
         }
      }
      strength.data[gindex1] = sqrt(pow(gradient_x, 2) + pow(gradient_y, 2));
      degrees = floor(((atan2(gradient_y, gradient_x) * 180 / PI)/45) + 0.5f);
      direction.data[gindex1] = (degrees == 4)?0:(degrees*45);
   }
}
```

インテル® ストリーミング SIMD 拡張命令 (インテル® SSE) の SIMD を利用した パフォーマンス

- 使用したコンパイラー・オプション: -O2 –qopt-report5 –qopt-report-phase=vec –std=c++11 –qopenmp-simd
- computeGradient の実行時間 = 0.106005 秒 (ベクトル化されていないバージョンよりも 3.19 倍高速)

OpenMP* の simd プラグマによる明示的なベクトル化の結果

General Exploration General Exploration viewpoint (change)

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Event Count | Platform

Grouping: Function / Call Stack

Function / Call Stack	Back-End Bound						Retiring
	Core Bound						
	Divider	Port Utilization					
		Cycles of 0 Ports Utilized	Cycles of 1 Port Utilized	Cycles of 2 Ports Utilized	Cycles of 3+ Ports Utilized	Vector Capacity Usage (FPU)	
computeGradient	23.2%	14.7%	48.4%	23.2%	12.6%	50.0%	16.8%
nonMaximumSupression	0.0%	0.0%	0.0%	0.0%	54.1%	0.0%	28.4%

Vector Capacity Usage (ベクトル処理機能の使用率) が向上しているが、さらに向上できる可能性がある

intel Software

OpenMP* の simd プラグマによる明示的なベクトル化の結果

Summary | Survey & Roofline | Refinement Reports

Some target modules do not contain debug information

Suggestion: enable debug information for relevant modules.

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	Why No Vectorization?	Vectorized Loops: Vector ISA	Efficiency	Gain ...	VL (V...	Instruction Set Analysis: Traits	Data Types
[loop in main at sample.cc:135]	1 Assumed de...	0.000s	0.001s	Scalar	vector dependence p...						
[loop in main at sample.cc:138]	1 Opportunity...	0.000s	0.001s	Scalar	outer loop was not a...						
[loop in main at sample.cc:147]		0.000s	0.008s	Scalar	outer loop was not a...						Float32
[loop in main at sample.cc:150]	1 Opportunity...	0.000s	0.001s	Scalar	outer loop was not a...						
[loop in main at sample.cc:159]		0.000s	0.029s	Scalar	inner loop was alrea...						Float32; Int32
[stack]		0.000s	0.043s	Function							
_Init_MKL_Loader		0.000s	0.001s	Function							Float32
[loop in main at sample.cc:159]	3 Vector regi...	0.000s	0.029s	Vectorized (Bo...		SSE2	~72%	2.87x	4	Divisions; Packs; Shift...	Float32; Float6
[loop in GaussianFilter at sample....		n/a	n/a	Scalar Complete...	vectorization possibl...						
[loop in GaussianFilter at sample....		n/a	n/a	Scalar Complete...	outer loop was not a...						
[loop in computeGradient]		n/a	n/a	Scalar Complete...							

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

File: sample.cc:159 main

Line	Source	Total Time	%	Loop/Function Time	%	Traits
149						
150	for(int i = 0; i < rows; i++)					
151	{					
152	int index1 = (i+1)*newcols+1;					
153	int index = i*cols;					
154	for(int j = 0; j < cols; j++)					
155	outputimage.data[index+j] = outputimage2.data[index1+j];	1.000ms				
156	}					
157	//Step 3 Gradient strength and direction					
158	timer_start2 = std::chrono::system_clock::now();					
159	computeGradient(outputimage2, gradientstrength, gradientdirection);	10.000ms		29.000ms		Divisions; Packs; Shifts; Shuffles; Square Roots; Type Conversions; Unpacks

[loop in main at sample.cc:159]
Vectorized SSE; SSE2 loop processes Float32; Float64; Int16; Int32; Int64; UInt32 data type(s) and
No loop transformations applied

[loop in main at sample.cc:159]
Scalar loop with instructions that use SSE; SSE2 registers. Not vectorized: inner loop was already
Loop was distributed, chunk 2

明示的なベクトル化適用後の最適化レポート

```
ループ開始 sample.cc(56,3)
    リマーク #15381: ベクトル化のサポート: ループ本体内でアラインされていないアクセスが使用されました。
    リマーク #15305: ベクトル化のサポート: ベクトル長 4
    リマーク #15309: ベクトル化のサポート: 正規化されたベクトル化のオーバーヘッド 0.221
    リマーク #15417: ベクトル化のサポート: 浮動小数点数をアップコンバートします (単精度から倍精度 1)。[ /usr/local/include/c++/7.1.0/cmath(418,14) ]
    リマーク #15417: ベクトル化のサポート: 浮動小数点数をアップコンバートします (単精度から倍精度 1)。[ /usr/local/include/c++/7.1.0/cmath(418,14) ]
    リマーク #15417: ベクトル化のサポート: 浮動小数点数をアップコンバートします (単精度から倍精度 1)。[ sample.cc(71,14) ]
    リマーク #15418: ベクトル化のサポート: 浮動小数点数をダウンコンバートします (倍精度から単精度 1)。[ sample.cc(50,2) ]
    リマーク #15301: OpenMP SIMD LOOP がベクトル化されました。
    リマーク #15450: マスクなし非アライン・ユニット・ストライド・ロード: 18
    リマーク #15451: マスクなし非アライン・ユニット・ストライド・ストア: 2
    リマーク #15475: --- ベクトルのコストサマリー開始 ---
    リマーク #15476: スカラーのコスト: 572
    リマーク #15477: ベクトルのコスト: 198.750
    リマーク #15478: スピードアップの期待値: 2.860
    リマーク #15482: ベクトル化された算術ライブラリーの呼び出し: 2
    リマーク #15486: 除算: 2
    リマーク #15487: 型変換: 24
    リマーク #15488: --- ベクトルのコストサマリー終了 ---
    ループ開始 sample.cc(61,4)
```

ループ中の型変換の数を減らす

```
#pragma omp parallel for private(gradient_x, gradient_y, degrees)
for(int i = 1; i < rows-1; i++){
   int index = i*cols+1;
   int gindex = (i-1)*gcols;
   #pragma omp simd private(gradient_x, gradient_y, degrees)
   for(int j = 0; j < cols-2; j++){
      int index1 = index+j;
      int gindex1 = gindex+j;
      gradient_x = gradient_y = 0.0f;
      for(int k1 = -1; k1 <= 1; k1++){
         int index2 = index1+(k1*cols);
         for(int k2 = -1; k2 <=1; k2++){
            gradient_x += gradientx[k1+1][k2+1] * input.data[index2+k2];
            gradient_y += gradienty[k1+1][k2+1] * input.data[index2+k2];
         }
      }
      strength.data[gindex1] = sqrtf(powf(gradient_x, 2) + powf(gradient_y, 2));
      degrees = floorf(((atan2(gradient_y, gradient_x) * 180 / PI)/45) + 0.5f);
      direction.data[gindex1] = (degrees == 4)?0:(degrees*45);
   }
}
```

デフォルトでは、数学関数 sqrt、pow、floor は引数として倍精度浮動小数点型を受け付けるが、オリジナルのデータは単精度浮動小数点型であるため、型変換が必要となりベクトル効率に影響する

```
ループ開始 sample.cc(56,3)
      リマーク #15381: ベクトル化のサポート: ループ本体内でアラインされていないアクセスが使用されました。
      リマーク #15305: ベクトル化のサポート: ベクトル長 4
      リマーク #15309: ベクトル化のサポート: 正規化されたベクトル化のオーバーヘッド 0.373
      リマーク #15301: OpenMP SIMD LOOP がベクトル化されました。
      リマーク #15450: マスクなし非アライン・ユニット・ストライド・ロード: 18
      リマーク #15451: マスクなし非アライン・ユニット・ストライド・ストア: 2
      リマーク #15475: --- ベクトルのコストサマリー開始 ---
      リマーク #15476: スカラーのコスト: 556
      リマーク #15477: ベクトルのコスト: 116.000
      リマーク #15478: スピードアップの期待値: 4.760
      リマーク #15482: ベクトル化された算術ライブラリーの呼び出し: 2
      リマーク #15486: 除算: 2
      リマーク #15487: 型変換: 20
      リマーク #15488: --- ベクトルのコストサマリー終了 ---
   ループ開始 sample.cc(61,4)
```

computeGradient の実行時間 = 0.871432 秒 (最初の実装よりも 3.87 倍高速)

ベクトル効率の改善

Summary | Survey & Roofline | Refinement Reports

INTEL ADVISOR 2018

Some target modules do not contain debug information

Suggestion: enable debug information for relevant modules.

1 of 2

ROOFLINE

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	Why No Vectorization?	Vectorized Loops: Vect...	Efficiency	Gain ...	VL (V...	Instruction Set Analysis: Traits	Data Ty
[loop in main at sample.cc:138]	1 Opportunity...	0.000s	0.001s	Scalar	outer loop was not a...						
[loop in main at sample.cc:147]		0.000s	0.007s	Scalar	outer loop was not a...						Float32
[loop in main at sample.cc:150]	1 Opportunity...	0.000s	0.001s	Scalar	outer loop was not a...						
[loop in main at sample.cc:159]		0.000s	0.024s	Scalar	inner loop was alrea...						Float32
[loop in __svml_atan2f4_h9]		0.000s	0.002s	Inside vectorized							Float32
[stack]		0.000s	0.038s	Function							
__Init_MKL_Loader		0.000s	0.001s	Function							
[loop in main at sample.cc:159]	**2 Data type c...**	**0.000s**	**0.024s**	**Vectorized (Bo...**		**SSE2**	**~100%**	**4.79x**	**4**	**Packs; Shifts; Shuffle...**	**Float32**
[loop in GaussianFilter at sample.cc:34]		n/a	n/a	Scalar Complete...	vectorization possibl...						
[loop in GaussianFilter at sample.cc:34]		n/a	n/a	Scalar Complete...	outer loop was not a...						
[loop in computeGradient]		n/a	n/a	Scalar Complete...							

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

All known issues with all possible recommendations: C++ / Fortran

Issue: Potential underutilization of FMA instructions

Your current hardware supports the AVX2 instruction set architecture (ISA), which enables the use of fused multiply-add (FMA) instructions. Improve performance by utilizing FMA instructions.

Recommendation: Target the AVX2 ISA

Confidence: Low

Although static analysis presumes the loop may benefit from FMA instructions available with the AVX2 ISA, no AVX2-specific code executed for this loop. To fix: Use the `xCORE-AVX2` compiler option to generate AVX2-specific code, or the `axCORE-AVX2` compiler option to enable multiple, feature-specific, auto-dispatch code generation, including AVX2.

Windows* OS	Linux* OS
/QxCORE-AVX2 or /QaxCORE-AVX2	-xCORE-AVX2 or -axCORE-AVX2

Read More:

- ax, Qax; x, Qx
- Code Generation Options in the Intel® C++ Compiler 16.0 User and Reference Guide
- Compiling for the Intel® Xeon Phi™ processor x200 and the Intel® AVX-512 ISA and Vectorization Resources for Intel® Advisor Users

intel Software

インテル® AVX の利用

Summary | Survey & Roofline | Refinement Reports

INTEL ADVISOR 20

Some target modules do not contain debug information

Suggestion: enable debug information for relevant modules.

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	Why No Vectorization?	Vectorized Loops: Vect...	Efficiency	Gain ...	VL (V...	Instruction Set Analysis: Traits	Data T
main		0.000s	0.016s	Function						Appr. Reciprocals(AVX)...	Float32
[loop in main at sample.cc:138]	1 Opportunity...	0.000s	0.001s	Scalar	outer loop was not a...						
[loop in main at sample.cc:147]		0.000s	0.003s	Scalar	outer loop was not a...						Float32
[loop in main at sample.cc:150]	1 Opportunity...	0.000s	0.001s	Scalar	outer loop was not a...						
[stack]		0.000s	0.016s	Function							
[loop in main at sample.cc:159]		0.000s	0.006s	Scalar	inner loop was alrea...					Appr. Reciprocals(AVX)...	Float32
[loop in main at sample.cc:147]	1 Data type co...	0.000s	0.003s	Scalar Versions	1 outer loop was not ...					FMA; Type Conversions	Float32
[loop in main at sample.cc:159]		0.000s	0.005s	Inside vectorized							
[loop in main at sample.cc:159]	1 Data type c...	0.000s	0.006s	Vectorized (Bo...		AVX2	~100%	8.83x	8	FMA; Permutes; Shuff...	Float3
[loop in GaussianFilter at sample.cc:34]		n/a	n/a	Scalar Complete...	vectorization possibl...						
[loop in GaussianFilter at sample.cc:34]		n/a	n/a	Scalar Complete...	outer loop was not a...						

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

File: sample.cc:159 main

Line	Source	Total Time	%	Loop/Function Time	%	Traits
152	int index1 = (i+1)*newcols+1;					
153	int index = i*cols;					
154	for(int j = 0; j < cols; j++)					
155	outputimage.data[index+j] = outputimage2.data[index1+j];	1.000ms				
156	}					
157	//Step 3 Gradient strength and direction					
158	timer_start3 = std::chrono::system_clock::now();					
159	computeGradient(outputimage2, gradientstrength, gradientdirection);	5.000ms		6.000ms		Appr. Reciprocals(AVX); Divisions; FMA; Permutes; Shuffles; Square Roots; Type Conversions

```
[loop in main at sample.cc:159]
  Vectorized AVX; AVX2; FMA loop processes Float32; Int32; Int64; UInt32 data type(s) and includes
  No loop transformations applied
[loop in main at sample.cc:159]
  Scalar loop with instructions that use AVX; FMA registers. Not vectorized: inner loop was alread
  Loop was distributed, chunk 2
[loop in main at sample.cc:159]
  Scalar loop
  No loop transformations applied
```


intel Software

インテル® AVX を利用したパフォーマンス

- 使用したコンパイラー・オプション:
  - -O2 -qopt-report5 -qopt-report-phase=vec -qopenmp-simd -xCORE-AVX2 -std=c++11
- computeGradient の実行時間 = 0.0217945 秒 (最初の実装よりも約 16 倍高速)
- これは、インテル® SSE と比較してベクトル長が 2 倍であること、そして FMA 命令のサポートによる。以下のソーベルフィルターの最内ループは、FMA 命令により利点が得られる。

```
for(int k2 = -1; k2 <=1; k2++){
    gradient_x += gradientx[k1+1][k2+1] * input.data[index2+k2];
    gradient_y += gradienty[k1+1][k2+1] * input.data[index2+k2];
}
```

高精度の除算命令の生成を無効にする

- 使用したコンパイラー・オプション:
  - -O2 -qopt-report5 -qopt-report-phase=vec -qopenmp-simd -xCORE-AVX2 -no-prec-div -std=c++11
- computeGradient の実行時間 = 0.0207084 秒 (最初の実装よりも 16.3 倍高速)
- 同じコードを以下のコンパイラー・オプションを使用して GCC 7.1 でビルドすると、OpenMP* 4.0 の SIMD プラグマを使用しているにもかかわらずループがベクトル化されない
  - -O2 -ffast-math -fopenmp-simd -march=haswell -std=c++11 -ftree-vectorize -fopt-info-vec
- computeGradient の実行時間 = 0.249924 秒

OpenMP* を利用したマルチスレッド化

```
#pragma omp parallel for private(gradient_x, gradient_y, degrees)
for(int i = 1; i < rows-1; i++){
   int index = i*cols+1;
   int gindex = (i-1)*gcols;
   #pragma omp simd private(gradient_x, gradient_y, degrees)
   for(int j = 0; j < cols-2; j++){
      int index1 = index+j;
      int gindex1 = gindex+j;
      gradient_x = gradient_y = 0.0f;
      for(int k1 = -1; k1 <= 1; k1++){
         int index2 = index1+(k1*cols);
         for(int k2 = -1; k2 <=1; k2++){
            gradient_x += gradientx[k1+1][k2+1] * input.data[index2+k2];
            gradient_y += gradienty[k1+1][k2+1] * input.data[index2+k2];
         }
      }
      strength.data[gindex1] = sqrtf(powf(gradient_x, 2) + powf(gradient_y, 2));
      degrees = floorf(((atan2(gradient_y, gradient_x) * 180 / PI)/45) + 0.5f);
      direction.data[gindex1] = (degrees == 4)?0:(degrees*45);
   }
}
```

- 使用したコンパイラー・オプション:
- -O2 -qopt-report5 -qopt-report-phase=vec -qopenmp-simd -xCORE-AVX2 -no-prec-div -qopenmp -std=c++11
- computeGradient の実行時間 = 0.0099909 秒 (最初の実装よりも約 34 倍高速)

インテル® VTune™ Amplifier の全般解析レポート

General Exploration General Exploration viewpoint (change)

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Event Count | Platform

Grouping: Function / Call Stack

Function / Call Stack	Clockticks ▼	Instructions Retired	CPI Rate	Front-End Bound		Bad Speculation	
				Front-End Latency	Front-End Bandwidth	Branch Mispredict	Machine Clears
nonMaximumSupression	236,600,000	146,650,000	1.613	0.0%	0.0%	52.0%	0.0%
computeGradient	99,750,000	141,400,000	0.705	23.1%	0.0%	0.0%	0.0%
main	72,800,000	201,950,000	0.360	0.0%	1.6%	0.0%	0.0%
GaussianFilter	52,500,000	110,250,000	0.476	0.0%	0.0%	0.0%	0.0%
__svml_atan2f8_l9	50,400,000	39,550,000	1.274	9.1%	11.4%	0.0%	9.1%
convertRGBtoGray	14,700,000	43,050,000	0.341	31.3%	0.0%	0.0%	0.0%
__intel_avx_rep_memset	3,150,000	0		0.0%	0.0%	0.0%	0.0%
std::atan2	1,400,000	350,000	4.000	0.0%	0.0%	0.0%	0.0%
__svml_atan2f8	700,000	0		0.0%	0.0%	0.0%	0.0%
__sti__$E	0	0	0.000	0.0%	0.0%	0.0%	0.0%

非最大抑制コード

```
for(int i = 0; i < row; i++)
{
      int index = i*col;
      for(int j = index; j < index+col; j++)
      {
            int index1 = j-col;
            int index2 = j+col;
            switch(direction.data[j])
            {
                      case 0: if(!(strength.data[j]>strength.data[index1] &&
strength.data[j]>strength.data[index2]))
                                    strength.data[j] = 0;
                                    break;
                      case 45: if(!(strength.data[j]>strength.data[index1-1] &&
strength.data[j]>strength.data[index2+1]))
                                      strength.data[j] = 0;
                                      break;
                      case 90: if(!(strength.data[j]>strength.data[index-1] &&
strength.data[j]>strength.data[index+1]))
                                      strength.data[j] = 0;
                                      break;
                      case 135: if(!(strength.data[j]>strength.data[index1+1] &&
strength.data[j]>strength.data[index2-1]))
                                        strength.data[j] = 0;
                                        break;
             }
      }
}
```

- 非最大抑制の実行時間 = 0.0601606 秒

コンパイラーは反復間の依存性を仮定

OFF Batch mode

Run Roofline

Collect

1. Survey Target

Collect

1.1 Find Trip Counts and FL...

Collect

Trip Counts

FLOPS

Mark Loops for Deeper Anal...

Select loops in the Survey Report for Dependencies and/or Memory Access Patterns analysis.

1 loop is marked

2.1 Check Dependencies

Collect

Some target modules do not contain debug information

Suggestion: enable debug information for relevant modules.

ROOFLINE

Function Call Sites and Loops		Vector Issues	Self Time	Total Time	Type
[loop in nonMaximumSupression at sample.cc:...	☑	1 Assumed dependency present	0.017s	0.017s	Scalar
[loop in computeGradient at sample.cc:57]	☐	1 Data type conversions present	0.008s	0.010s	Vectorized (Body)
GaussianFilter	☐		0.005s	0.005s	Inlined Function
[loop in main at sample.cc:147]	☐	1 Assumed dependency present	0.002s	0.002s	Scalar
__svml_atan2f8_l9	☐		0.002s	0.002s	Vector Function
convertRGBtoGray	☐		0.001s	0.001s	Inlined Function
[loop in main at sample.cc:159]	☐	1 Assumed dependency present	0.001s	0.001s	Scalar
memmove	☐		0.001s	0.001s	Function
[loop in memset]	☐		0.001s	0.001s	Scalar
[loop in strlen]	☐		0.001s	0.001s	Scalar
[stack]	☐		0.000s	0.095s	Function
_start	☐		0.000s	0.087s	Function

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

File: sample.cc:86 nonMaximumSupression

Line	Source
76	return;
77	}
78	
79	void nonMaximumSupression(Mat &strength, Mat &direction)

インテル® Advisor の依存性解析

Summary | Survey & Roofline | Refinement Reports

Site Location	Loop-Carried Dependencies	Strides Distributio
[loop in nonMaximumSupression at sample.cc: ..	RAW:1	No information ava

```
84      {
85          int index = i*col;
86          for(int j = index; j < index+col; j++)
87          {
88              int index1 = j-col;
```

Memory Access Patterns Report | Dependencies Report | Recommendations

Problems and Messages

ID		Type	Site Name	Sources	Modules	State
P1		Parallel site information	loop_site_162	sample.cc	sample	✓ Not a problem

intel Software

ベクトル依存関係を無視する

```
for(int i = 0; i < row; i++)
{
      int index = i*col;
      #pragma ivdep
      for(int j = index; j < index+col; j++)
      {
            int index1 = j-col;
            int index2 = j+col;
            switch(direction.data[j])
            {
                        case 0: if(!(strength.data[j]>strength.data[index1] &&
strength.data[j]>strength.data[index2]))
                                    strength.data[j] = 0;
                                    break;
                        case 45: if(!(strength.data[j]>strength.data[index1-1] &&
strength.data[j]>strength.data[index2+1]))
                                     strength.data[j] = 0;
                                     break;
                        case 90: if(!(strength.data[j]>strength.data[index-1] &&
strength.data[j]>strength.data[index+1]))
                                     strength.data[j] = 0;
                                     break;
                        case 135: if(!(strength.data[j]>strength.data[index1+1] &&
strength.data[j]>strength.data[index2-1]))
                                       strength.data[j] = 0;
                                       break;
             }
      }
}
```

- 非最大抑制の実行時間 = 0.00604505 秒 (最初の実装よりも 9.95 倍高速)
- ループのデータ型が unsigned char であることを考慮すると、理想的なスピードアップは 32 倍

ベクトル化により分岐予測ミスを排除

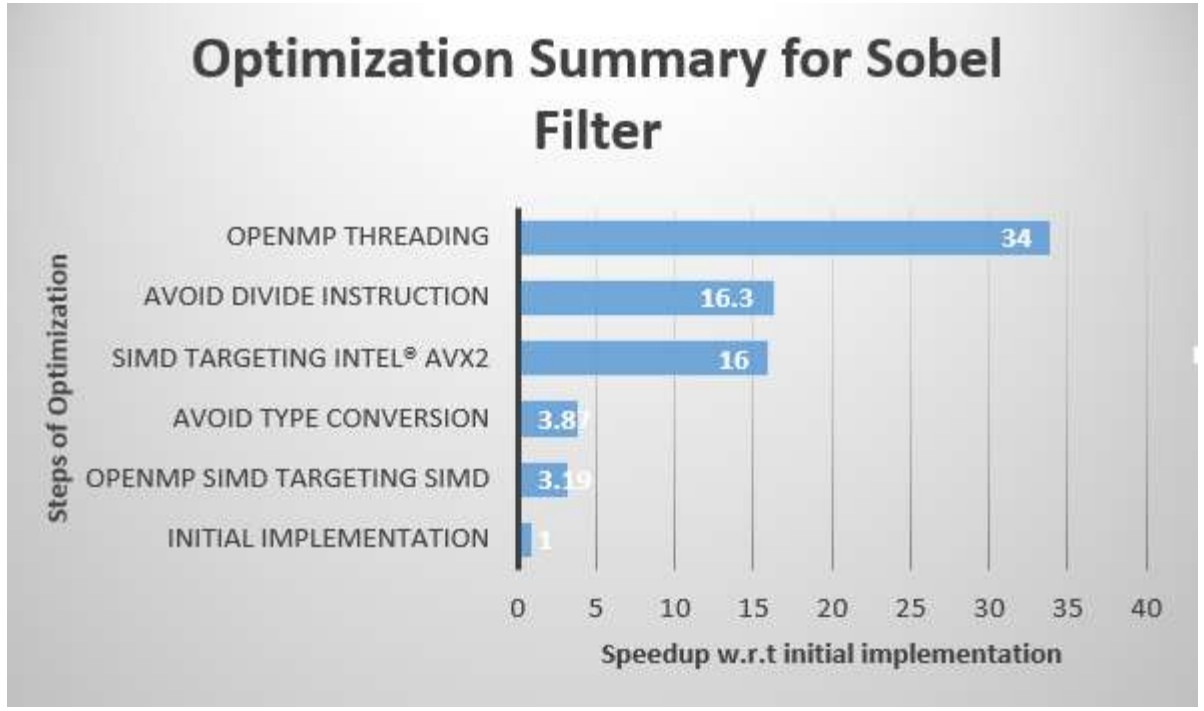
General Exploration General Exploration viewpoint (change)

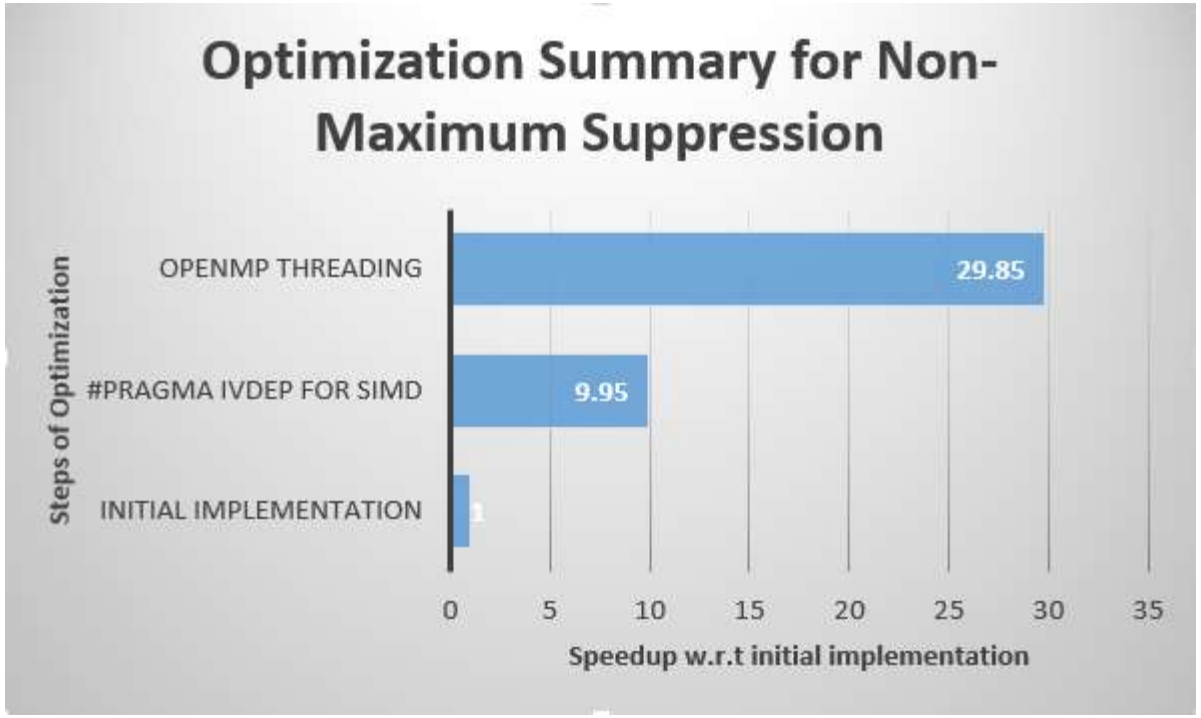
Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Event Count | Platform

Grouping: Function / Call Stack

Function / Call Stack	Clockticks ▼	Instructions Retired	CPI Rate	Front-End Bound		Bad Speculation	
				Front-End Latency	Front-End Bandwidth	Branch Mispredict	Machine Clears
computeGradient	103,250,000	152,600,000	0.677	26.7%	0.0%	0.0%	0.0%
main	66,850,000	201,600,000	0.332	0.0%	0.0%	0.0%	0.0%
GaussianFilter	55,650,000	111,650,000	0.498	0.0%	0.0%	0.0%	0.0%
__svml_atan2f8_l9	42,700,000	27,650,000	1.544	0.0%	21.5%	0.0%	21.5%
nonMaximumSupressionompparallel_for@83	35,000,000	49,000,000	0.714	0.0%	0.0%	0.0%	0.0%
convertRGBtoGray	16,800,000	42,350,000	0.397	0.0%	0.0%	0.0%	0.0%
__intel_avx_rep_memset	2,450,000	350,000	7.000	0.0%	0.0%	0.0%	0.0%
std::atan2	700,000	700,000	1.000	0.0%	0.0%	0.0%	0.0%
__svml_atan2f8	350,000	350,000	1.000	0.0%	0.0%	0.0%	0.0%
cv::Mat::release	0	0	0.000	0.0%	0.0%	0.0%	0.0%

サマリー





次のステップ

インテル® Parallel Studio XE 2018 をお試しください!

- 製品の評価版: https://www.isus.jp/intel-parallel-studio-xe/

ウェビナーもご覧ください!

- https://software.seek.intel.com/fall-webinar-series (英語)
  - **メモリー・アクセス・プロファイル: 一般的なパフォーマンス・ボトルネックの特定と解決方法** (英語) – **オンデマンド**
- https://software.intel.com/en-us/articles/webinar-better-threaded-performance-and-scalability-with-intelr-vtune-amplifier-openmp (英語)

法務上の注意書きと最適化に関する注意事項

本資料の情報は、現状のまま提供され、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も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、商品性に関する保証、第三者の特許権、著作権、その他、知的財産権の侵害への保証を含む) をするものではありません。

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。

最適化に関する注意事項

インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。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® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサー用のものがあります。この注意事項で言及した命令セットの詳細については、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。

注意事項の改訂 #20110804

intel®

Software