インテル® Advisor を使用して パフォーマンスを読み解こう

整数ルーフライン編

入門編

2019 年 8 月
iSUS 編集長
すがわら　きよふみ

目標

インテル® Advisor 2019 update4 の機能を利用して、
アプリケーションのパフォーマンスを理解します

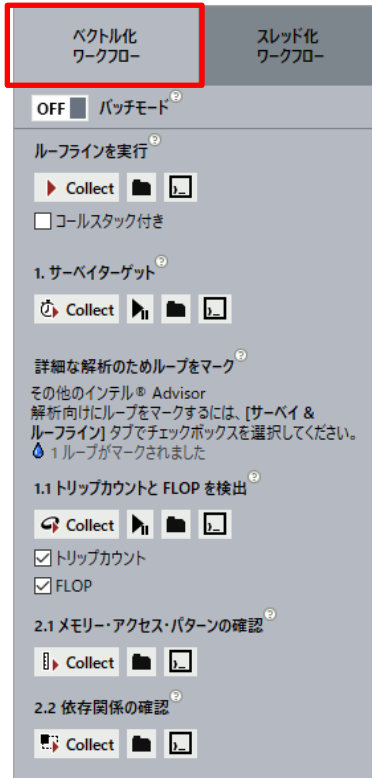
注目する機能:

- 統合ルーフライン (浮動小数点、整数、浮動小数点 + 整数)

ルーフラインに関する詳細:
https://www.isus.jp/products/advisor/roofline_webinar/

© 2019 iSUS *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。

インテル® Advisor 2019 概要



ベクトル化解析:

- **ルーフライン**は、アプリケーションの計算とメモリー帯域幅の上限を視覚化できるようにデータをグラフ化します
- **サーベイ**は、最も時間を費やしているループと詳細な SIMD 統計を表示します
- **トリップカウントと FLOPS** は、各ループと関数の反復回数、呼び出し回数、および正確なFLOPS を測定します
- **メモリー・アクセス・パターン解析**は、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します
- **依存性解析**は、ループにベクトル化や並列化の妨げとなる反復間の依存性がないか、動的な依存性解析を行います

© 2019 iSUS *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。

update4 の新機能

- [サマリー] ペインで、ループおよびアプリケーションごとの推奨事項が表示されるようになりました。これは、Update3 で削除された「上位 5 つの推奨事項」に代わるもので、これを拡張したものです
- [サーベイ] と [コード] 解析は、ニューラル・ネットワーク・アプリケーションで使用される VNNI (開発コード名 Cascade Lake でサポート) を使用するループ/関数を識別できるようになりました
- FLOPS とトリップカウント収集は、既存のデータを置き換えるのではなく拡張するようになりました。例えば、トリップカウントが収集され、その後 FLOPS のみを収集するため 2 回目の解析が行われた場合、FLOPS データはトリップカウントを置き換えることなく結果に追加されます

詳細は、https://www.isus.jp/products/advisor/advisor-2019-overview/

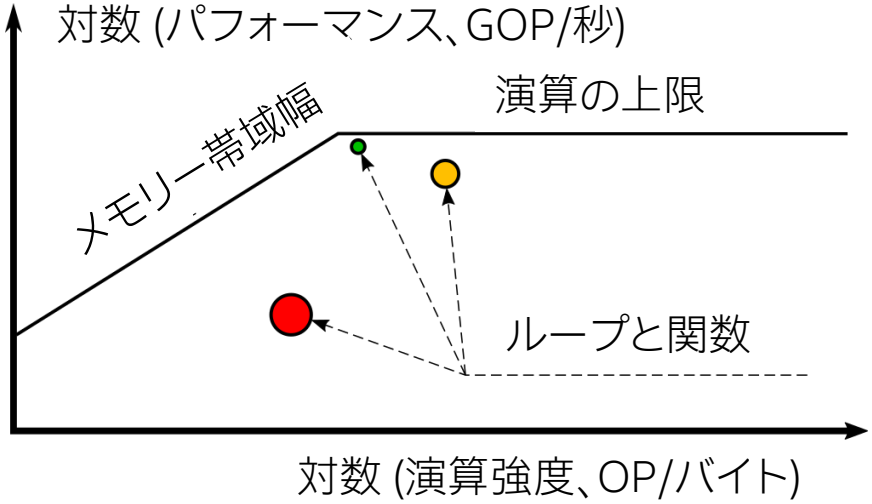
© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

統合ルーフライン
(浮動小数点、整数、浮動小数点 + 整数)

© 2019 iSUS *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。

整数ルーフライン

メモリー帯域幅や計算ピークを含む、ハードウェアのパフォーマンスの上限に関連したアプリケーションのパフォーマンスを視覚的に表現する機能



インテル® Advisor は、情報を収集してインタラクティブなルーフライン・グラフに自動的に配置

ADVIXE_EXPERIMENTAL=intops_strict
これにより、ADD、MUL、IDIV、および SUB 操作のみがカウントされます。

© 2019 iSUS *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。

整数ルーフライン・モデルの基本

メトリック

インテル® Advisor の整数ルーフラインは 2 つのメトリックをベースにしています。
縦軸はパフォーマンスを示します。1 秒あたりの INT 操作の数 (INTOPS) として計算される単純なメトリックです。横軸は演算強度 (AI) を示します

整数演算

整数ルーフラインは継続して改良が行われており、INT 操作の正確な定義は変わる可能性があります。現在は、ADD、ADC、SUB、MUL、IMUL、DIV、IDIV、INC/DEC、シフト、ローテートなどが、INT 操作としてカウントされます。

演算強度

演算強度は、メモリーからリードされるデータと、そのデータに対する算術演算の関係を示します。アプリケーションが、合計 *A* バイトのメモリーをリードして、*B* のスカラー算術演算を行った場合、演算強度は、演算強度 = *B/A* として計算できます

© 2019 iSUS *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。

演算強度の例

アプリケーションが、2 つの 32 ビット整数 (a と b) をリードし、*a + b* を計算する場合:

```
1  MOV eax, a       // 4 バイトリード
2  MOV eax, b       // 4 バイトリード
3  ADD eax, ebx     // INTOP: eax = a + b
```

演算強度 =
1 int op (整数操作) / (2 メモリー操作 * 4 バイト)
= 0.125 INTOP/バイト

アプリケーションが、3 つの 32 ビット整数 (a、b および c) をリードし、*(a + c) * (a + b)* を計算する場合:

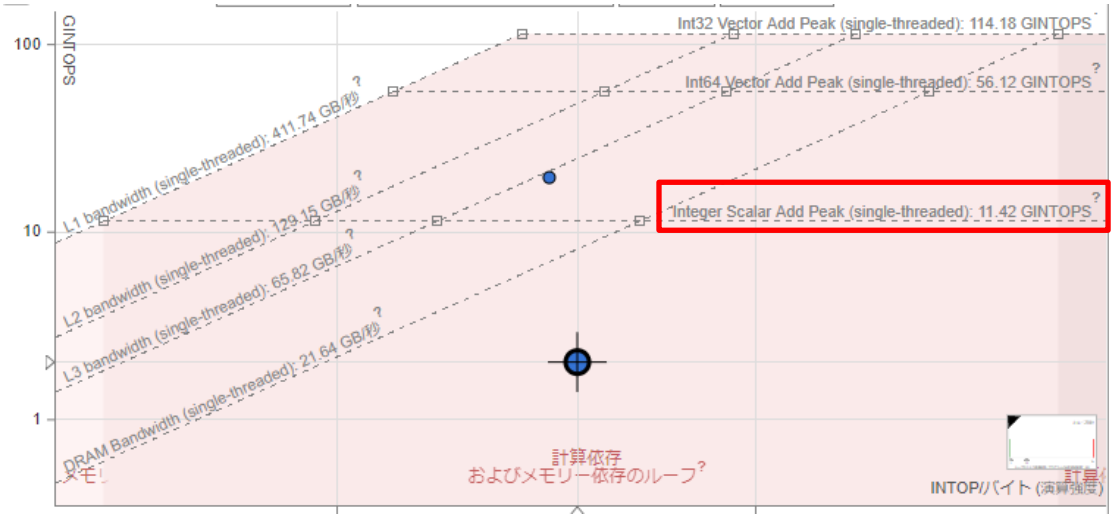
```
1  MOV eax, a       // 4 バイトリード
2  MOV eax, b       // 4 バイトリード
3  ADD ebx, eax     // INTOP: ebx = a + b
4  MOV eax, c       // 4 バイトリード
5  ADD eax, ebx     // INTOP: eax = a + c
6  MUL ebx          // INTOP: eax = (a + c) * (a + b)
```

演算強度 =
3 int op (整数操作) / (3 メモリー操作 * 4 バイト)
= 0.25 INTOP/バイト

© 2019 iSUS *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。

ルーフのドットとライン

インテル® Advisor のルーフライン･モデルは、アプリケーションを解析する前にいくつかのベンチマークを実行し、さまざまなレベルのメモリー階層の帯域幅や計算上の制限など、システム･ハードウェアの上限を表示します



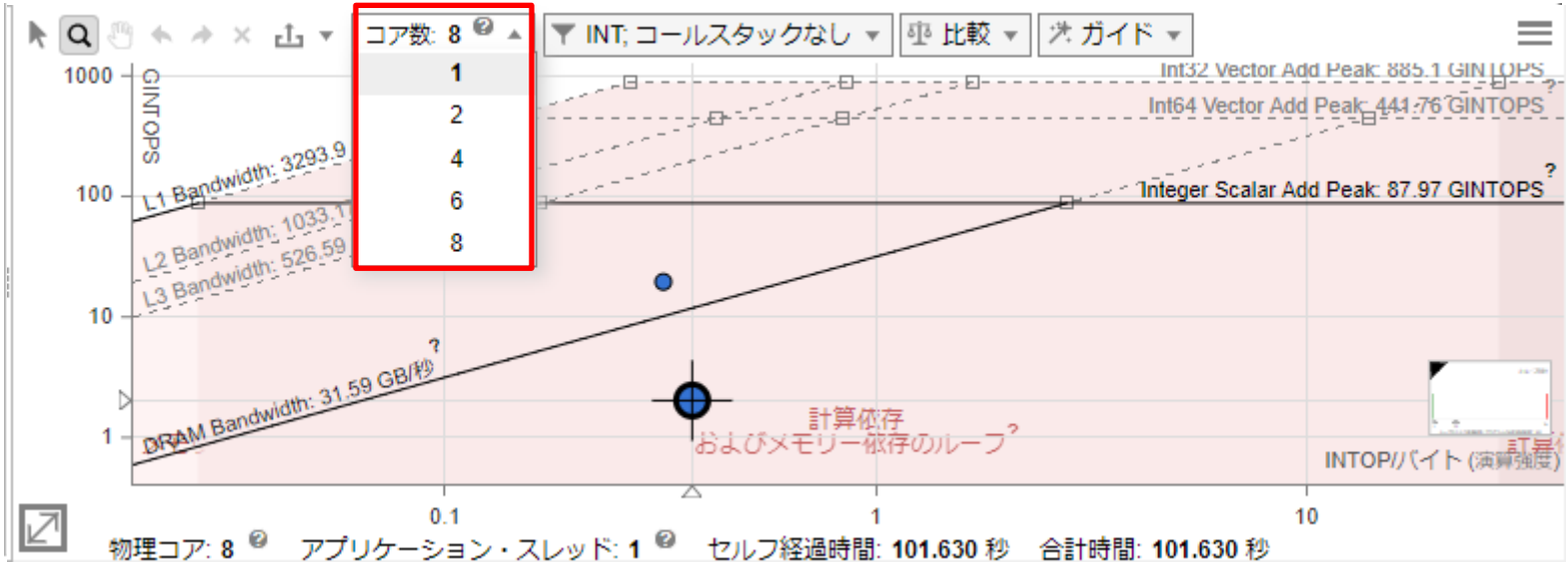
Z390 + DDR4 (2666MHz) * 2 + Core i7 9700K

上限データは、ルーフとしてグラフ上に表示され、最適化なしでマシンが提供できる最大パフォーマンスを示します

例えば、スカラー加算ピークは、ベクトル化なしで 1 秒間に処理できる加算演算の最大数を表します

© 2019 iSUS *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。

ループのデータとコア数

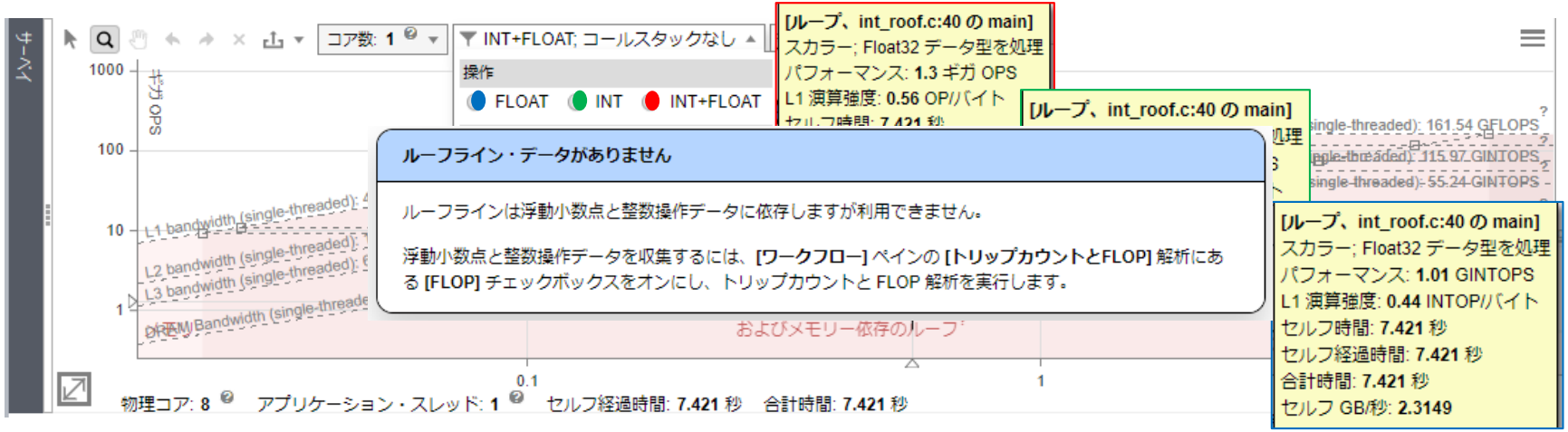


ユーザーは、必要に応じて GUI のドロップダウン・メニューから手動でスケーリングを設定できます

© 2019 iSUS *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。

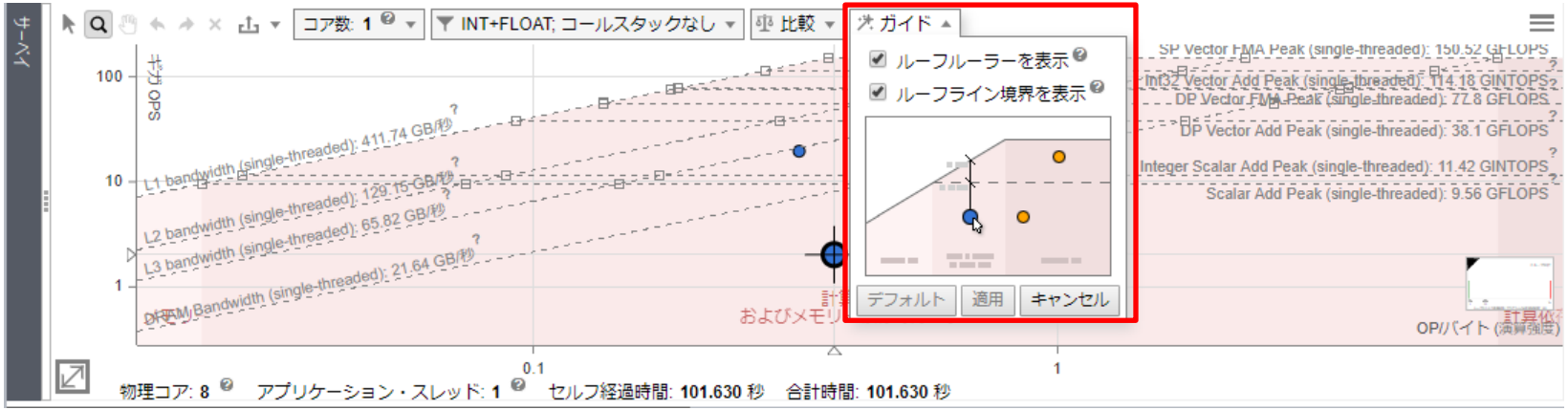
整数ルーフライン解析を実行

ルーフライン･データの収集は、関連する計算の種類にかかわらず行われます。ルーフライン･グラフ上部のドロップダウン･メニューから、[FLOAT]、[INT] または [INT+FLOAT] データ表示に切り替えることができます



© 2019 iSUS *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。

ルーラーと境界を表示



ルーフルーラー: 選択したループ/関数から最も近いパフォーマンス･ルーフまでの垂直線を表示します。ルーフと交差するポイントには、ハードウェアの上限を示すラベルが示されます

ルーフライン境界: ルーフラインのゾーンを色分けして、ループ/関数がメモリー、計算または計算とメモリーのいずれに依存するかを確認しやすくします

© 2019 iSUS *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。

整数ループラインの最適化: ソースコード

シンプルな 2 次元配列の行列乗算と加算

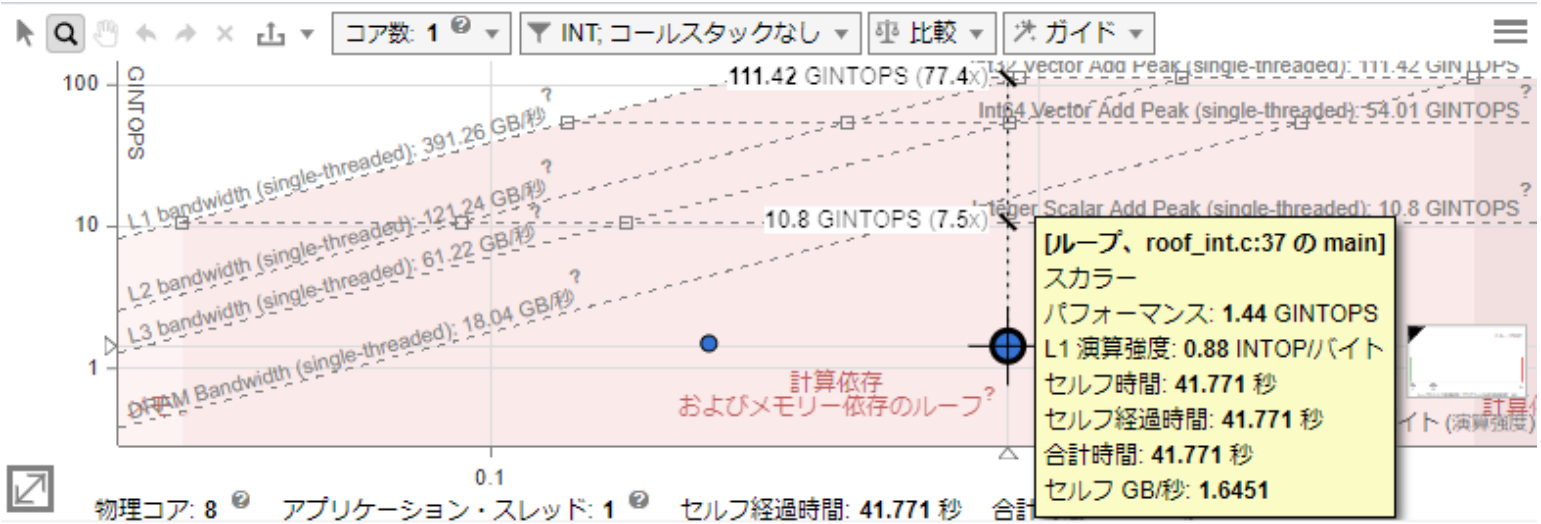
```
for(i=0; i<m; i++)
    for(j=0; j<p; j++)
        for(k=0; k<n; k++)
            c[i][j] += a[i][k] * b[k][j];
```

- 配列 a、b、c は int 型の 2 次元配列
- それぞれの次元のサイズは 2048 (4 バイト x 2048 x 2048 = 16M バイト)
- 全体のデータサイズはおよそ 48M バイト

© 2019 iSUS *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。

整数ルーフラインの最適化:

行列乗算加算を最適化



- このサンプル･アプリケーションは、ほぼ乗算加算で構成されているため、ループを表すドットは大きく表示され、最適化の過程を通してこの傾向は変わりません
- 実際のプログラムの一部として乗算加算が実行される場合、全体の実行時間が短くなるのに比例して、ドットは最適化の過程で小さくなっていくと予想されます

© 2019 iSUS *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。

メトリックの計算

7/(4*2)=0.875 INTOP/バイト
8,589,934,592 * 8 バイト=68,719,486,736= 68.719 GB
8,589,934,592 * 7 = 60,129,542,144 = 60.130 GINTOP
60.130 GINTOP / 41.771 秒 = 1.439 GINTOPS

```
30  ⊞  for(i=0; i<m; i++)
31  ⊞    for(j=0; j<p; j++)
32  ⊞      for(k=0; k<n; k++){
33            c[i][j] += a[i][k] * b[k][j];
34          }
```

1.136s	1.410s	
0.274s		

アドレス	行	アセンブリー	合計時間	%	セルフ時間	%
0x140006d3a		**Block 1: 1073741824**				
0x140006d3a	33	mov r9d, dword ptr [rsi+r10*1+0x6ef620]				
0x140006d42	33	lea r8, ptr [rsi+r15*4+0xaef620]	274.282ms		274.282ms	
0x140006d4a	33	imul r9d, dword ptr [r11+r8*1]				
0x140006d4f	32	inc r12	965.064ms		965.064ms	
0x140006d52	33	add r13d, r9d				
0x140006d55	32	add r11, 0x1000	170.874ms		170.874ms	
0x140006d5c	32	add r10, 0x4				
0x140006d60	32	cmp r12, rax				
0x140006d63	32	jl 0x140006d3a <Block 1>				

4バイトリード

計算

7 OP / 8 バイト
= 0.875

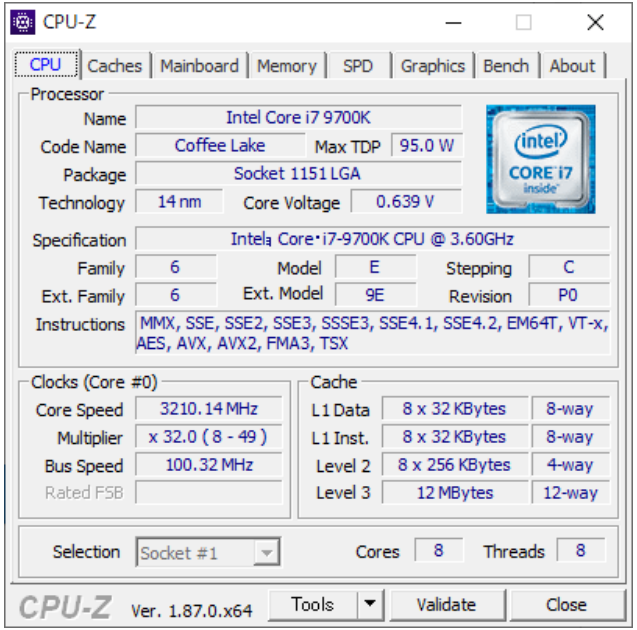
関数呼び出しサイトとループ	CPU 時間				計算パフォーマンス			メモリー		トリップカウント	
	合計時間	セルフ時間	合計経過...	セルフ経過時間	セルフ GINTOPS	セルフ INT AI	セルフ GINTOP	セルフメモリー (GB)	セルフメモリー (GB/秒)	平均	呼び出し数
[ループ、roof_int.c:37 の main]	41.771s	41.771s	41.771秒	41.771秒	1.439	0.875	60.130	68.719	1.645	2048	4194304

© 2019 iSUS *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。

インテル® Advisor 2019 の利用例 (coffeeLake-Ref)

最適化オプションによるパフォーマンスの違いをインテル® Advisor 2019 で調査

最適化オプション	実行時間
/Zi /O1	41.771 秒
/Zi /O2	
/Zi /QxHOST (AVX2)	
/Zi /QxHOST /Qopenmp (ブロッキング & AVX2)	



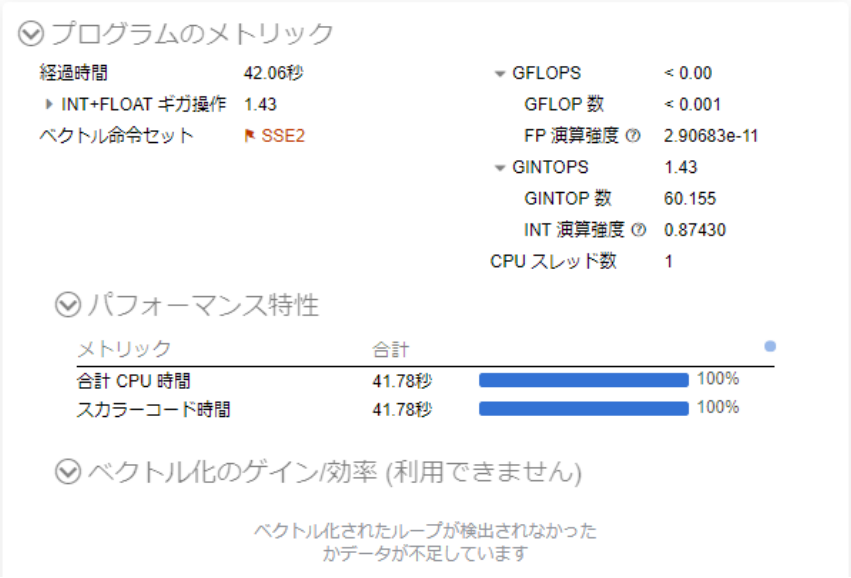
© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

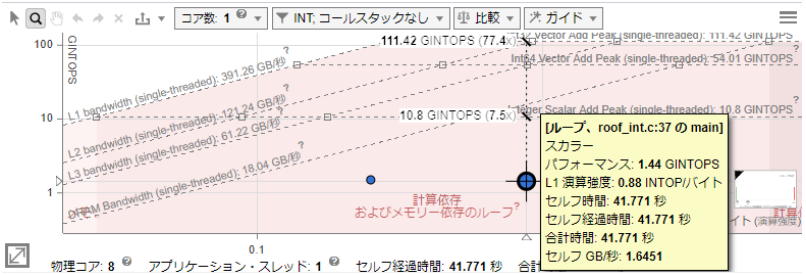
ループを最適化

ステップ 1 : ベクトル化
ステップ 2 : 最適化オプションを適用 (ベクトル化、行列変換)
ステップ 3 : 最新の命令セットを使用
ステップ 4 : キャシュブロッキング
ステップ 5 : データのアライメント

© 2019 iSUS *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。

ステップ 1: ベクトル化





- ベクトル命令セットには、SSE2 が利用されていますが、ループはベクトル化されておらず、非効率なメモリーアクセスがあります

関数呼び出しサイトとループ	パフォーマンス問題	タイプ
[ループ、roof_int.c:32 の main]	2 非効率なメモリー・アクセス・パターンがあります	スカラー

- /O1 ではベクトル化は有効にならない

コンパイラー・オプション: /nologo /Zi /debug:inline-debug-info /Qopt-report:3 /Qopt-report-phase:vec,loop /O1 /Feroof_int_o1 /Qopt-report-file:roof_int_o1.optrpt

© 2019 iSUS *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。

ステップ 1：推奨事項を確認

ソース | トップダウン | コード解析 | アセンブリー | 推奨事項 | ベクトル化できない理由

インテル® Advisor が検出できるすべての問題 C++ | Fortran

非効率なメモリー・アクセス・パターンがあります

不規則な (可変またはランダム) ストライドアクセスのメモリーアクセス命令の比率が高いことが観測されました。調査して適切に処理することでパフォーマンスを改善します。

AoS の代わりに SoA を使用します

配列は、通常のインデックスでアクセス可能な、連続したデータ項目の集合を含む最も一般的なデータ構造です。構造体配列 (AoS)、または配列構造体 (SoA) として構成することができます。AoS はカプセル化には最適ですが、効率良いベクトル処理の妨げになることがあります。解決方法: AoS の代わりに SoA を使用するデータ構成にコードを変更します。

SDLT を使用します

AoS から SoA への移行コストがパフォーマンスの利点を上回る可能性があります。解決方法: インテル® C++ コンパイラー 16.1 で導入された SIMD Data Layout Templates (SDLT) を使用して、移行のコストを軽減できます。SDLT は、C++11 テンプレート・ライブラリーであり、コードの変更が数行で済みます。

例 (元のコード)

```
...
struct kValues {
    float Kx;
    float Ky;
    float Kz;
    float PhiMag;
};
std::vector<kValues> dataset(count);
...
```

FMA 命令が効率良く使用されていない可能性があります

現在のハードウェアは、乗算加算融合 (FMA) 命令が利用できるインテル® AVX2 命令セット・アーキテクチャー (ISA) をサポートします。FMA 命令を使用してパフォーマンスを改善します。

インテル® AVX2 ISA ターゲット

静的解析は、ループがインテル® AVX2 ISA 以降で使用可能な FMA 命令から利点が得られると推測しましたが、このループではFMA 命令が実行されていません。解決方法: 次のコンパイラー・オプションを使用します。

- `xCORE-AVX2` を使用してインテル® AVX2 をサポートするマシンとしないマシン向けにコンパイルします。
- `axCORE-AVX2` を使用してインテル® AVX2 をサポートするマシン向けにコンパイルします。
- `xCOMMON-AVX512` を使用してインテル® AVX-512 をサポートするマシン向けにコンパイルします。
- `axCOMMON-AVX512` を使用してインテル® AVX-512 をサポートするマシンとしないマシン向けにコンパイルします。

注: コンパイラー・オプションは CPU マイクロアーキテクチャーに依存します。

- 最適化オプションを変更することで、ループ変換による最適化が行われる可能性があります:
  /O2 または /O3
- 命令セットを指定することを検討します

© 2019 iSUS *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。

ステップ 2 : 最適化オプションを変更 /O2

コンパイラー・オプション: /nologo /Zi /debug:inline-debug-info /Qopt-report:3 /Qopt-report-phase:vec,loop /O2 /Feroof_int_o1 /Qopt-report-file:roof_int_o1.optrpt

プログラムのメトリック

経過時間	42.06秒	GFLOPS	< 0.00
INT+FLOAT ギガ操作	1.43	GFLOP 数	< 0.001
ベクトル命令セット	SSE2	FP 演算強度	2.90683e-11
		GINTOPS	1.43
		GINTOP 数	60.155
		INT 演算強度	0.87430
		CPU スレッド数	1

パフォーマンス特性

メトリック	合計	
合計 CPU 時間	41.78秒	100%
スカラーコード時間	41.78秒	100%

ベクトル化のゲイン/効率 (利用できません)

ベクトル化されたループが検出されなかったかデータが不足しています

プログラムのメトリック

経過時間	2.33秒	GFLOPS	< 0.00
INT+FLOAT ギガ操作	11.99	GFLOP 数	< 0.001
ベクトル命令セット	SSE2, SSE	FP 演算強度	1.93742e-11
		GINTOPS	11.99
		CPU スレッド数	1

パフォーマンス特性

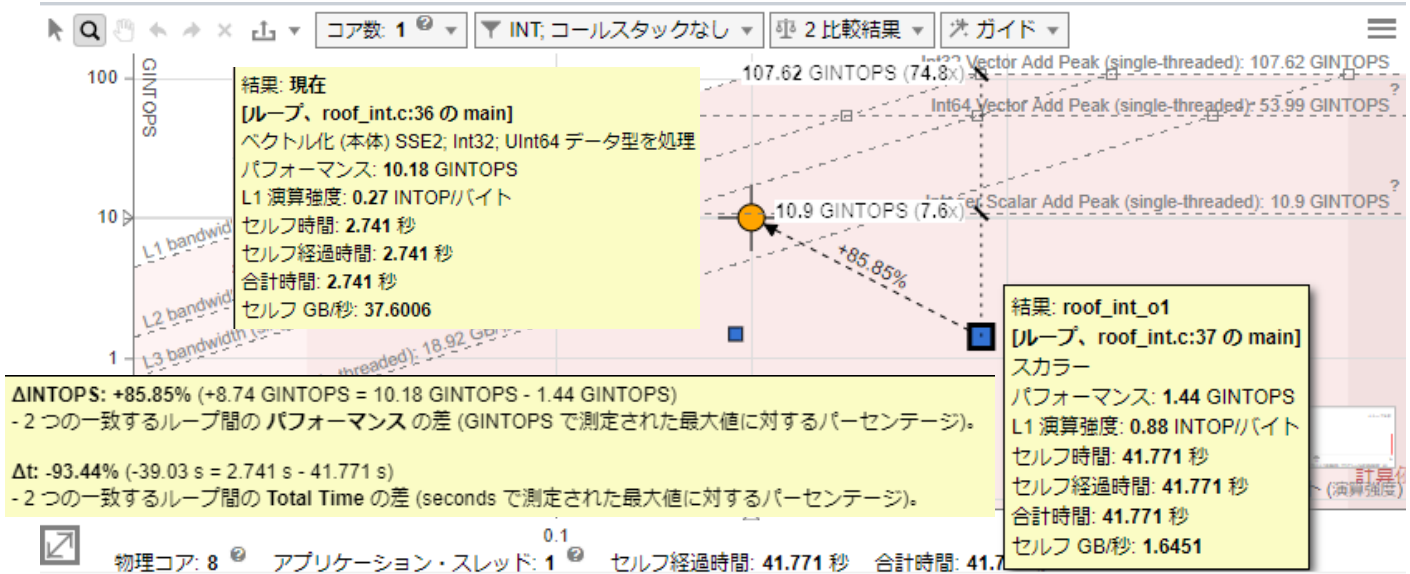
メトリック	合計	
合計 CPU 時間	2.30秒	100%
2 要素でベクトル化されたループの時間	2.28秒	99.3%
スカラーコード時間	0.02秒	0.7%

ベクトル化のゲイン/効率

ベクトルループのゲイン/効率	3.55x	89%
プログラムのおよそのゲイン	3.53x	

© 2019 iSUS *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。

ステップ 2 : 最適化オプションを変更 /O2



- DRAM の帯域幅を超え、
- L3 のルーフに迫り、
- スカラーピーク加算ループを上回る

命令セット:	パフォーマンス:	演算強度:	セルフ経過時間:
スカラー	1.44 GINTOPS	0.88 INTOP/バイト	41.711 秒
↓	↓	↓	↓
ベクトル	10.18 GINTOPS	0.27 INTOP/バイト	2.741 秒

© 2019 iSUS *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。

ステップ 2 : 行列変換 (/O2 による最適化)

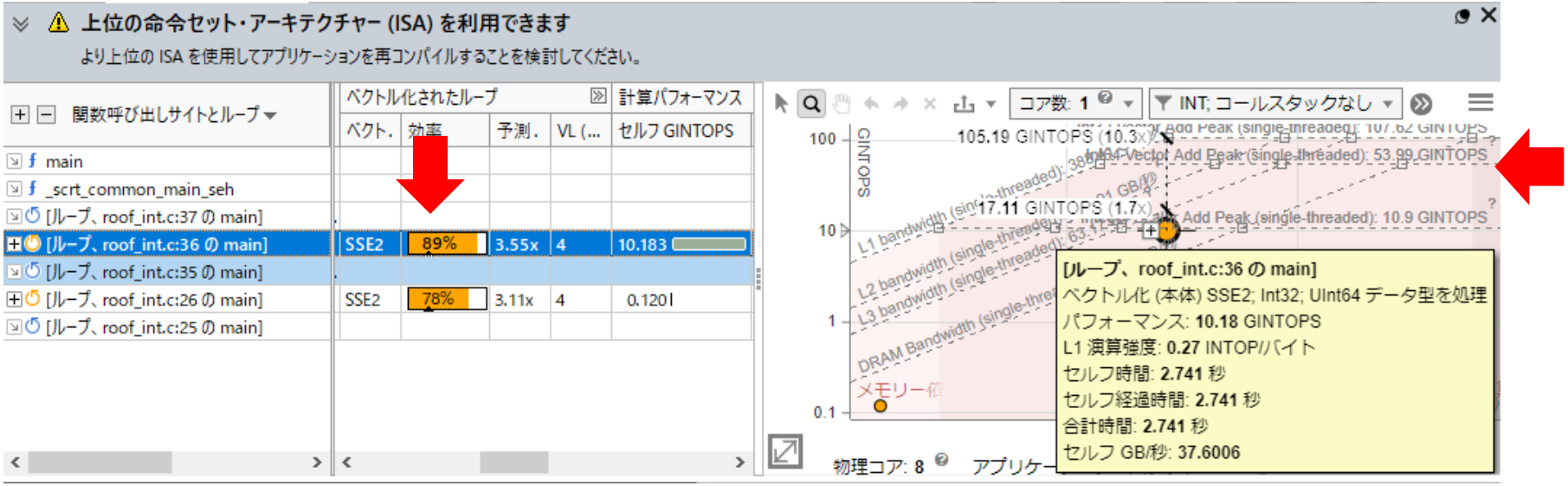
関数呼び出しサイトとループ	CPU 時間 合計時間	CPU 時間 セルフ時間	タイプ	ベクトル化できない理由	ベクトル化されたループ ベクト.	ベクトル化されたループ 効率	ベクトル化されたループ 予測.	ベクトル化されたループ VL (...	計算パフォーマンス セルフ GINTOPS	計算パフォーマンス セルフ INT AI	メモリー	トリップカウント 平均	トリップカウント 呼び出し数	命令セット解析 特性	命令セット解析 データ型	命令セット解析 ベクト.	詳細
[ループ、roof_int.c:36 の main]	2.741s	2.741s	ベクトル化 (本体)		SSE2	89%	3.55x	4	10.183	0.271	103.079	128	4194304	Shifts	Int32; U...	6	アンロール by 4; ...
[ループ、roof_int.c:36 の main]	2.741s 99.8%	2.741s 99.8%	ベクトル化 (本体)		SSE2			4	10.183	0.271	103.079	128	4194304	Shifts	Int32; U...	6	アンロール by 4; ...
[ループ、roof_int.c:35 の main]	2.741s 99.8%	0.000s	スカラー	loop was not vect...						0.063	< 0.001	2048	1				

行	Source	合計時間	%	ループ/関数時間	%	特性
35	for(i=0; i<m; i++)					
	[ループ、roof_int.c:35 の main] スカラー ループ. Loop was not vectorized: inner loop was already vectorized ループ変換は適用されていません					
36	for(j=0; j<p; j++)	0.031s		2.741s		
	[ループ、roof_int.c:36 の main] ベクトル化 SSE2 ループ Int32; UInt64 データ型を処理 and Shifts を含む Loop was interchanged; loop was unrolled by 4 [ループ、roof_int.c:36 の main] ベクトル化 SSE2 リマインダー・ループ [未実行] Int32; UInt64 データ型を処理 and Shifts を含む Loop was interchanged [ループ、roof_int.c:36 の main] スカラー リマインダー・ループ [未実行] Loop was interchanged					
37	for(k=0; k<n; k++){					
	[ループ、roof_int.c:37 の main] スカラー ループ SSE2 レジスターを使用する命令を含む. Loop was not vectorized: inner loop was already vectorized Loop was interchanged					
38	c[i][j] += a[i][k] * b[k][j];	2.711s				Shifts
39	}					

それぞれのループに
適用された最適化

© 2019 iSUS *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。

ステップ 3: 最新の命令セットを使用



ベクトル化されているにもかかわらず、ループはベクトル加算ピークからは離れています

SSE2 は、スカラーに対して整数ループが最大 4 倍高速化される利点がありますが、このマシンで利用可能な最上位の AVX2 命令セットでは、その 2 倍のパフォーマンスを提供できます

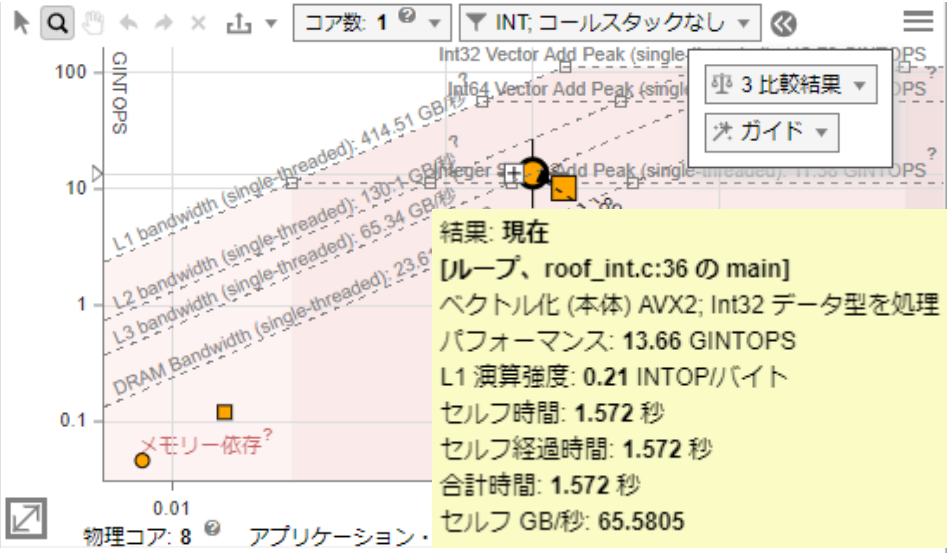
© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ステップ 3: AVX2 命令セットを使用

関数呼び出しサイトとループ	ベクトル化されたループ				計算パフォーマン
	ベクト.	効率	予測.	VL (...	セルフ GINTOPS
[ループ、roof_int.c:36 の main]	AVX2	100%	10.30x	8	13.663
[ループ、roof_int.c:36 の main]	AVX2			8	13.663
[ループ、roof_int.c:26 の main]	AVX	84%	6.69x	8	0.046
_scrt_common_main_seh					

ΔINTOPS: +25.48% (+3.48 GINTOPS = 13.66 GINTOPS - 10.18 GINTOPS)
- 2 つの一致するループ間の **パフォーマンス** の差 (GINTOPS で測定された最大値に対するパーセンテージ)。

Δt: -42.65% (-1.169 s = 1.572 s - 2.741 s)
- 2 つの一致するループ間の **Total Time** の差 (seconds で測定された最大値に対するパーセンテージ)。



パフォーマンス:
10.18 GINTOPS
↓
13.66 GINTOPS

演算強度:
0.27 INTOP/バイト
↓
0.21 INTOP/バイト

セルフ経過時間:
2.741 秒
↓
1.572 秒

© 2019 iSUS *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。

ステップ 4 : キャシュブロッキング

次に調査する必要があるメモリー帯域幅について考えてみましょう

MAP 解析は、メモリー使用量がまだ理想からはかけ離れていることを示していますが、ここでは、それほどストライドは離れていません

サイトの位置	ループ伝搬依存	ストライド分散	アクセスパターン	容量の予測	
				命令ごとの最大アドレス範囲	最初のインスタンスのサイト容量
[ループは main にあります: roof_int.c ..	利用可能な情報がありません	50% / 50% / 0%	混在したストライド	16MB	48MB
[ループは main にあります: roof_int.c ..	利用可能な情報がありません	100% / 0% / 0%	すべてユニットストライド	7KB	15KB
[ループは main にあります: roof_int.c ..	利用可能な情報がありません	75% / 25% / 0%	混在したストライド	16MB	16MB

```
35      for(i=0; i<m; i++)
36        for(j=0; j<p; j++)
37          for(k=0; k<n; k++){
38            c[i][j] += a[i][k] * b[k][j];
39        }
```

メモリー・アクセス・パターン・レポート | 依存関係レポート | 推奨事項

ID		ストライド	タイプ	ソース	入れ子関数	変数参照	命令ごと
P2		256	一定ストライド	roof_int.c:38		b	16MB
P6			並列サイト情報	roof_int.c:37			
P11		0	均一ストライド	roof_int.c:38		c	7KB
P13		0	均一ストライド	roof_int.c:38		c	7KB
P15		1	ユニットストライド	roof_int.c:38		a	6KB

100%: ユニットストライドやストライド 0 アクセスの
メモリー命令の比率
ユニットストライド (ストライド 1) = 命令は反復間で 1 要素分離れた
メモリーにアクセスします
ユニフォーム・ストライド (ストライド 0) = 命令は反復間で同じ
メモリーにアクセスします
0%: 固定または定数の非ユニット・ストライド・アクセスの
メモリー命令の比率
定数ストライド (ストライド N) = 命令は反復間で N 要素分離れた
メモリーにアクセスします
例: 倍精度浮動小数点型の場合、ストライド 4 はこの命令が
アクセスするメモリーアドレスは、反復ごとに 32 バイト (4*sizeof(double))
増加することを意味します
0%: 不規則な (可変またはランダム) ストライドアクセスの
メモリー命令の比率
不規則なストライド = 命令は反復間で予測できない要素数分離れた
メモリーにアクセスします
一般に、配列の間接インデックス・アクセスで見られます
例: a[index[i]]
- ギャザー (不規則) アクセス (インテル® AVX2 命令セット・アーキテクチャーの v(p)gather* 命令を検出します)
- スキャッター (不規則) アクセス (インテル® AVX2 命令セット・アーキテクチャーの v(p)scatter* 命令を検出します)

© 2019 iSUS *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。

メモリー・アクセス・パターンの詳細

ファイル: roof_int.c:38

行	Source	ストライド	オペランド型	ベクトル長	オペランドサイズ (ビット)	ループ・インスタンスの量
31	}					
32						
33	start = clock();					
34						
35	for(i=0; i<m; i++)					
36	for(j=0; j<p; j++)					
37	for(k=0; k<n; k++){					
38	c[i][j] += a[i][k] * b[k][j];	[1]; [1]	int32	8	256	7KB
39	}					
40						
41	finish = clock();					
42	walltime = (double)(finish - start)/CLOCKS_PER_SEC;					
43	printf("¥n Matmul int kernel wall clock time = %f sec.¥n", walltime					
44						

詳細ビュー

ユニットストライド (ストライド 1)

オペランドサイズ (ビット): 256
オペランド型: int32
ベクトル長: 8
メモリーアクセス量: 7KB

変数参照

ユニットストライド (ストライド 1)

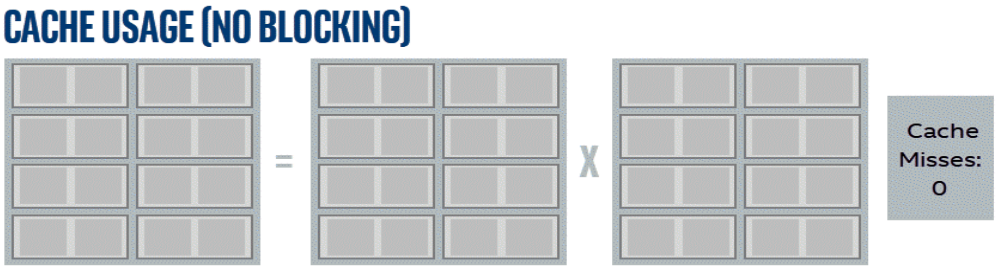
モジュール: roof_int_host.exe!0x140006c10

アドレス	行	アセンブリー	物理ストライド	オペランド情報	ベクトル長	オペランドサイズ (ビット)	アドレス範囲	メモリーアクセス量
0x140006e60	38	vpbroadcastd ymm0, xmm0						
0x140006e65	36	xor r11d, r11d						
0x140006e68		**Block 24:**						
0x140006e68	38	lea r12, ptr [rbp+r11*4+0x1a22960]						
0x140006e70	38	vpmulld ymm1, ymm0, ymmword ptr [r12+r10*1]	1	int32*8	8	256	0x2a20980 ...	7KB
0x140006e76	38	lea r14, ptr [rax+r11*4]						
0x140006e7a	38	vpaddd ymm2, ymm1, ymmword ptr [rbp+r14*1+0x2a229	1	int32*8	8	256	0x3a20980 ...	7KB
0x140006e84	36	add r11, 0x8						
0x140006e88	38	vmovdqu ymmword ptr [rbp+r14*1+0x2a22960], ymm2	1	int32*8	8	256	0x3a20980 ...	7KB
0x140006e92	36	cmp r11, r8						
0x140006e95	36	jb 0x140006e68 <Block 24>						
0x140006e97		**Block 25:**						
0x140006e97	36	cmp rsi, rcx						
0x140006e9a	36	inb 0x140006ec0 <Block 27>						

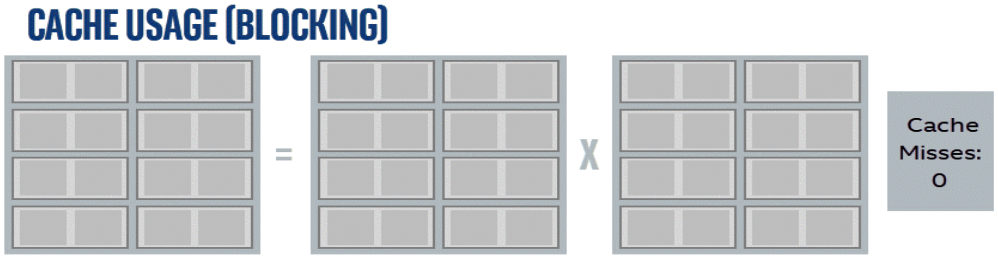
命令ごとのメモリー・アクセス・パターン

© 2019 iSUS *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。

ブロッキングあり/なし



この例では、8 個の小さなキャッシュラインを保持できる縮小キャッシュを使用して、4x4 行列を乗算し、各ラインは 2 つの要素を保持しています。この例の行は、参照されてから時間が経過すると排出されます



あるレベルのキャッシュ階層に収めてデータをタイル全体で計算できるように結果行列を分割すると、1 度に 1 つのタイルを計算することで、キャッシュミスを最小限に抑えることができます。アニメーションの例でも、この手法の利点が分かります

© 2019 iSUS *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。

ステップ 4 : キャシュブロッキング

あるレベルのキャッシュ階層に収めてデータをタイル全体で計算できるように結果行列を分割すると、1 度に 1 つのタイルを計算することで、キャッシュミスを最小限に抑えることができます

```
for(ichunk = 0; ichunk < m; ichunk += CHUNK_SIZE){
    for(jchunk = 0; jchunk < m; jchunk += CHUNK_SIZE){
        for(i = 0; i < CHUNK_SIZE; i++){
            ci = ichunk + i;
            for(j = 0; j < CHUNK_SIZE; j++){
                cj = jchunk + j;
#pragma omp simd reduction(+:c[ci][cj])
                for(k = 0; k < n; k++){
                    c[ci][cj] += a[ci][k] * t[cj][k];
}   }   }   }   }
```

CHUNK_SIZE は、256K L2 キャッシュ向けに 16 に設定

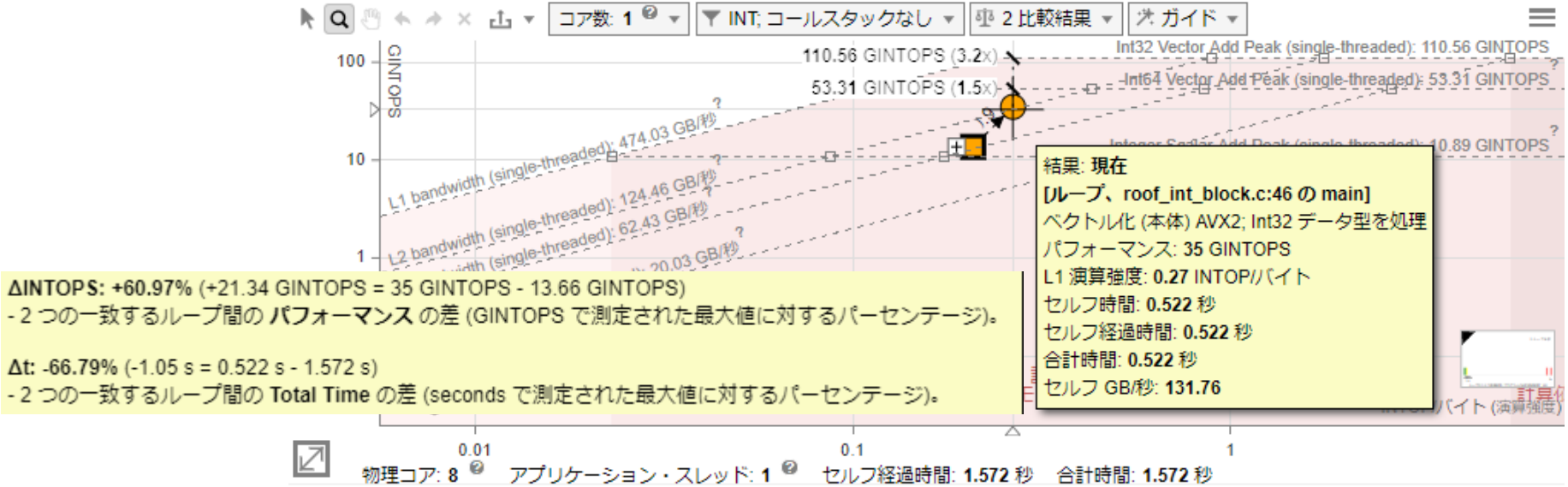
キャッシュサイズ >= データ型サイズ * (N^2 + 2 * (N * マトリクスの幅))

262,144 バイト >= 4 バイト * (N^2 + 2 * (N * 2048))

131072 >= N^2 + 4096N

© 2019 iSUS *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。

ステップ 4 : キャシュブロッキング



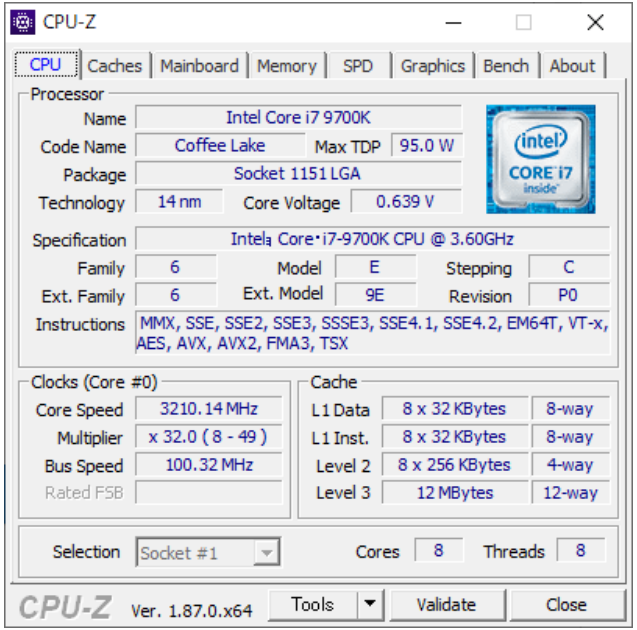
パフォーマンス:	演算強度:	セルフ経過時間:
13.66 GINTOPS	0.21 INTOP/バイト	1.572 秒
↓	↓	↓
35.00 GINTOPS	0.27 INTOP/バイト	0.522 秒

© 2019 iSUS *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。

インテル® Advisor 2019 の利用例 (coffeeLake-Ref)

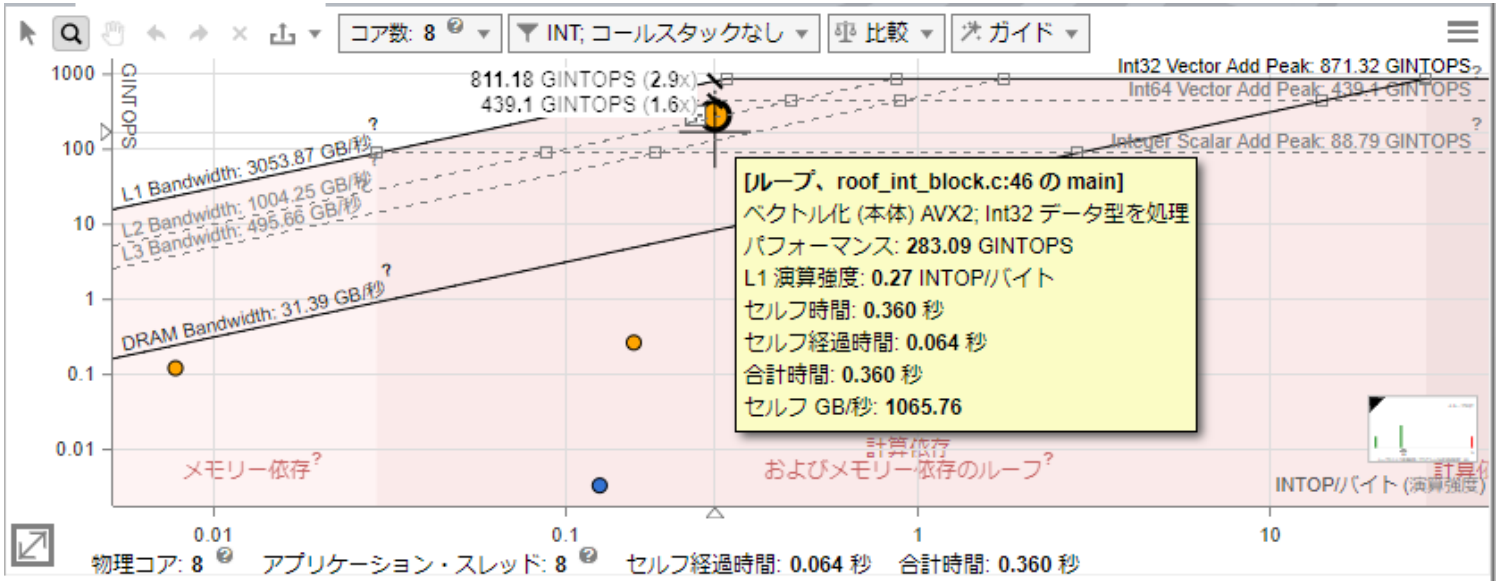
最適化オプションによるパフォーマンスの違いをインテル® Advisor 2019 で調査

最適化オプション	実行時間
/Zi /O1	41.771 秒
/Zi /O2	2.741 秒
/Zi /QxHOST (AVX2)	1.572 秒
/Zi /QxHOST /Qopenmp (ブロッキング & AVX2)	0.522 秒



© 2019 iSUS *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。

さらに … /Qparallel オプションで並列化



パフォーマンス:
35.00 GINTOPS
↓
283.09 GINTOPS

演算強度:
0.27 INTOP/バイト
↓
0.27 INTOP/バイト

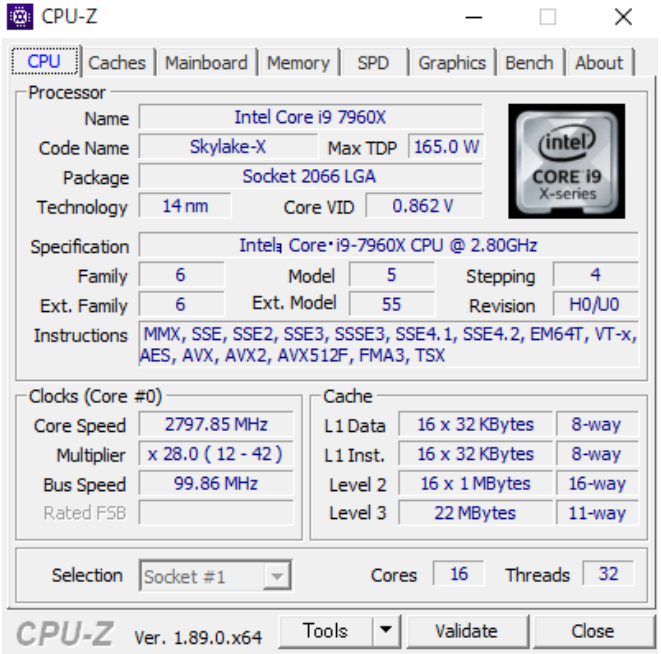
セルフ経過時間:
0.522 秒
↓
0.064 秒

© 2019 iSUS *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。

インテル® Advisor 2019 の利用例 (Skylake-X)

最適化オプションによるパフォーマンスの違いをインテル® Advisor 2019 で調査

最適化オプション	実行時間
/Zi /O1	35.855 秒
/Zi /O2	2.170 秒
/Zi /QxHOST (AVX512)	1.077 秒
/Zi /QxHOST /Qopenmp (ブロッキング 16)	0.515 秒
/Zi /QxHOST /Qopenmp (ブロッキング 32)	0.482 秒
/Zi /QxHOST /Qopenmp (ブロッキング 64)	0.467 秒
/Zi /QxHOST /Qopenmp (ブロッキング 128)	0.541 秒



© 2019 iSUS *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。

まとめ

- インテル® Advisor 2019 を使用して、最適化の検証、目標値の理解、改善によるゲインの確認を容易に行えます
- インテル® Advisor 2019 が最適化の作業効率を大幅に改善します

© 2019 iSUS *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。

サンプルコードと環境

このセッションで使用したサンプルコードのダウンロード:
http://www.isus.jp/wp-content/uploads/code/webinar/roof_int_sample.zip

名前	種類
makefile	ファイル
roof_int	C Source
roof_int_block	C Source

zip ファイルを展開すると 3 つのファイルがあります

makefile は、Windows* の Visual Studio* 環境のインテル® C++ コンパイラー向けに記述されています。ソース自体は、Linux* や macOS* 環境の C コンパイラーでコンパイルできます。

使用したコンパイラーとツール:
インテル® C++ コンパイラー V19 update4
インテル® Advisor 2019 update4 (日本語パッケージ適用)

© 2019 iSUS *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。

インテル® ソフトウェア開発ツール

日本語版パッケージを提供中

- インテル® Advisor 2018 update 1, 2, 3
- インテル® Advisor 2019 Initial, update 1, 2, 4
  https://www.isus.jp/products/advisor/advisor_jp/

- インテル® VTune™ Amplifier 2019 Initial, update 1, 2, 3, 4, 5, 6
  https://www.isus.jp/products/vtune/vtune_jp/

インテル® 64 アーキテクチャーおよび IA-32 アーキテクチャー最適化リファレンス・マニュアル (参考訳最新 041 版)

公開中のインテル® Parallel Universe マガジン日本語版

© 2019 iSUS *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。

参考資料

インテル® Advisor 2019 の新機能
https://www.isus.jp/products/advisor/advisor-2019-overview/

インテル® Advisor を使用してメモリーのボトルネックを排除
https://www.isus.jp/products/advisor/remove-memory-bottlenecks-using-advisor/

インテル® Advisor の整数ルーフライン・モデル
https://www.isus.jp/products/advisor/brief-overview-of-integer-roofline-modeling-in-advisor/

インテル® Advisor 導入ガイド
https://www.isus.jp/products/advisor/advisor-get-started/

ベクトル化の効率を把握する
https://www.isus.jp/products/psxe/pu36-04-how-effective-is-your-vectorization/

© 2019 iSUS *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。

iSUS