インテル® Advisor 2019 の新機能
日本語版の計画

2018 年 10 月
iSUS 編集長
すがわら　きよふみ

はじめに

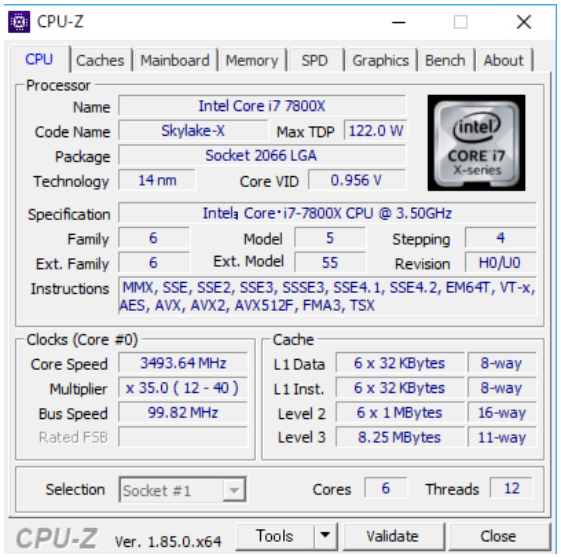
- iSUS では IA 向けのソフトウェア開発者を支援するため、多くの記事や技術情報、および製品ドキュメントの日本語化を行ってきました
- インテル® コンパイラー製品は、インテル社による日本語版コンポーネントが提供されているため、それ以外のツールの日本語化の要望をあげてきましたが、なかなか製品化されません
- そこで、iSUS では独自にパフォーマンス・ツールの日本語化を計画し、今回インテル® Advisor 2018 の日本語版を提供できるようになりました
- 現在インテル® Advisor 2019 とインテル® VTune™ Amplifier 2019の日本語版を開発中です

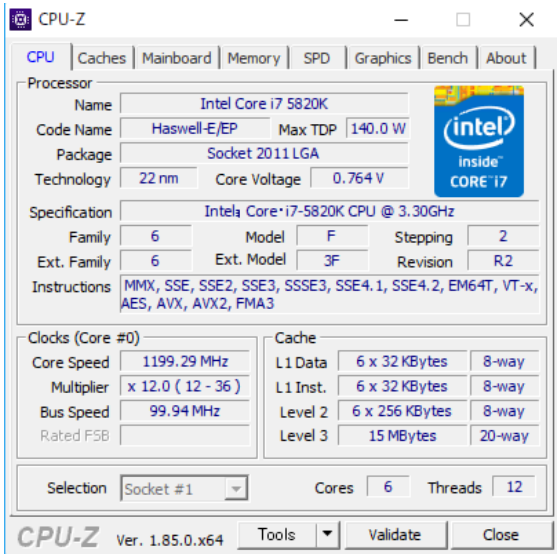
内容

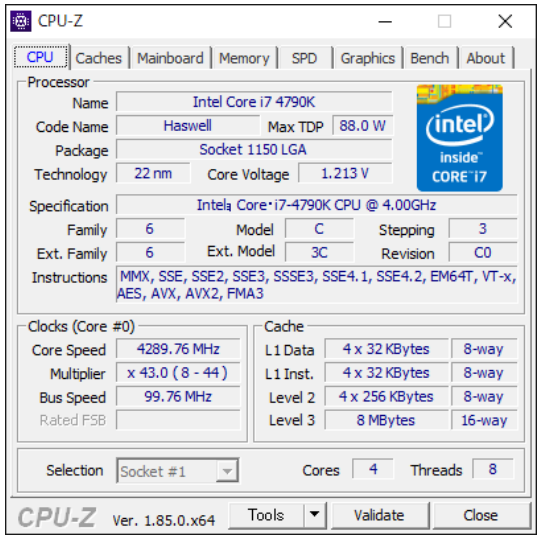
- インテル® Advisor 2019 の新機能
  - 整数ルーフライン解析を使用した整数計算の最適化
  - 正確なメモリー使用量を取得し、キャッシュ・シミュレーションを使用して複数のハードウェア構成をチェック
  - Linux* や Windows* 上で収集されたデータを macOS* ユーザー・インターフェイスで表示して解析
  - フローグラフ・アナライザーによるグラフ・アルゴリズムのプロトタイプ作成
  - 新しい推奨事項: C++ と Parallel STL の標準アルゴリズムを最適化

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

始める前に：CPU をチェックします

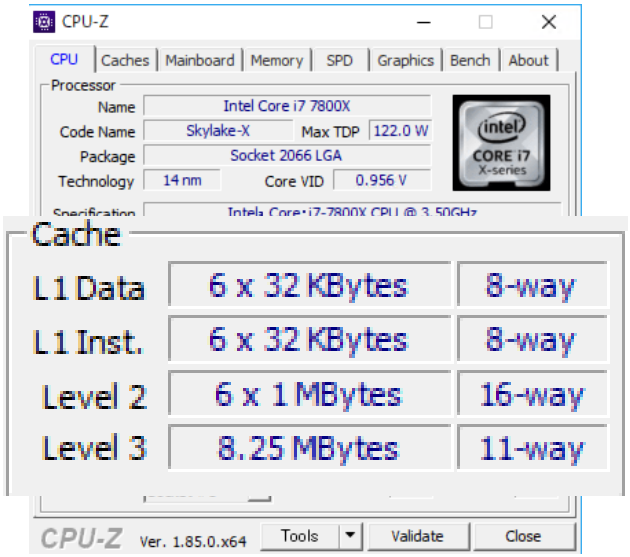


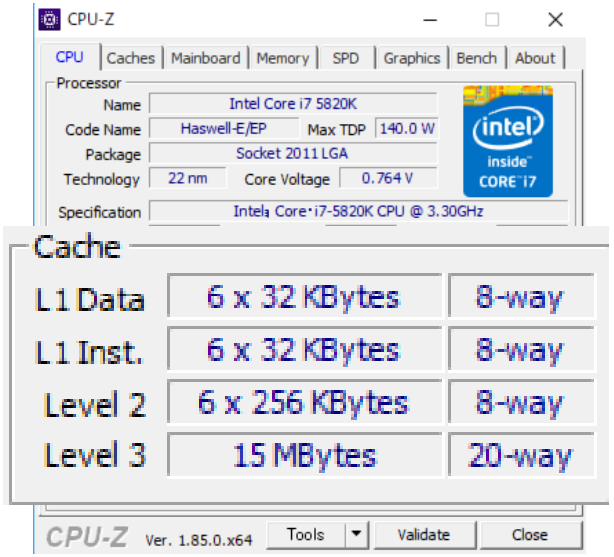


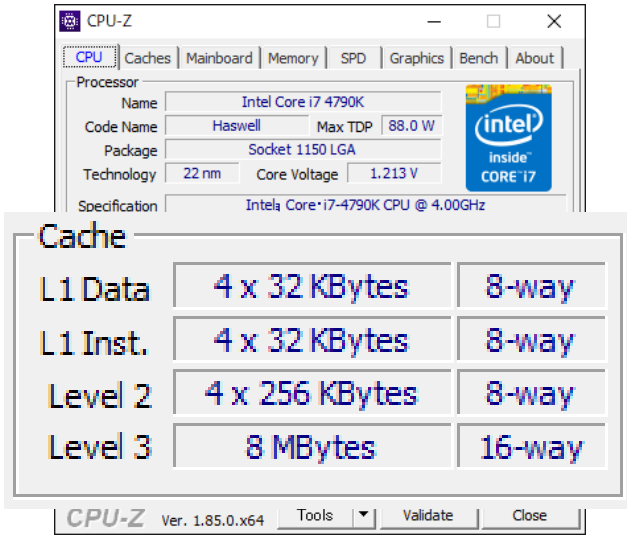


インテル® ソフトウェア開発ツールと共に提供される cpuinfo も利用できます

始める前に : CPU をチェックします





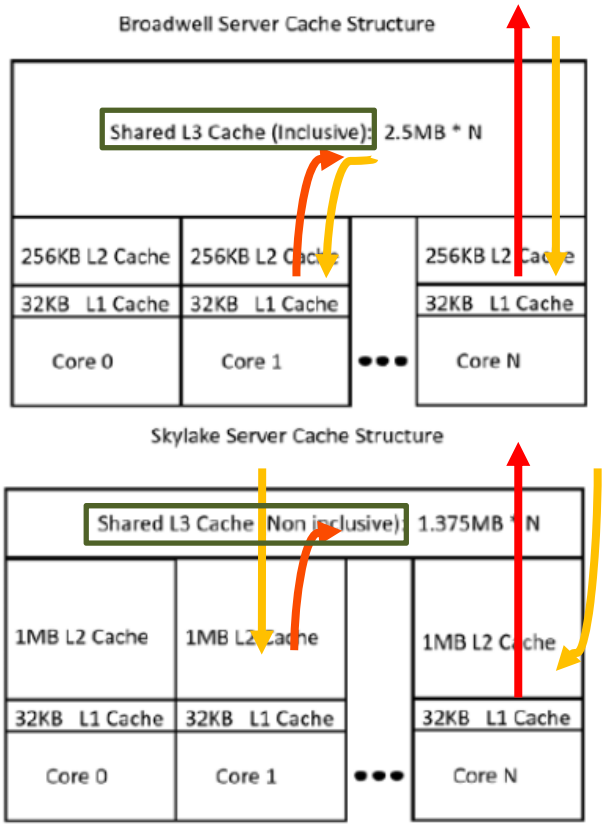


インテル® ソフトウェア開発ツールと共に提供される cpuinfo も利用できます

インテル® マイクロアーキテクチャー Skylake Server† のキャッシュ

キャッシュレベル	カテゴリー	Broadwell† マイクロアーキテクチャー	Skylake Server† マイクロアーキテクチャー
L1 データ･キャッシュ･ユニット (DCU)	サイズ [KB]	32	32
	レイテンシー [サイクル]	4-6	4-6
	最大帯域幅 [バイト/サイクル]	96	192
	持続帯域幅 [バイト/サイクル]	93	133
	連想性 [ウェイ]	8	8
L2 中間キャッシュ (MLC)	サイズ [KB]	256	1024 (1MB)
	レイテンシー [サイクル]	12	14
	最大帯域幅 [バイト/サイクル]	32	64
	持続帯域幅 [バイト/サイクル]	25	52
	連想性 [ウェイ]	8	16
L3 ラストレベルキャッシュ (LLC)	サイズ [MB]	最大 2.5	最大 1.375[1]
	レイテンシー [サイクル]	50-60	50-70
	最大帯域幅 [バイト/サイクル]	16	16
	持続帯域幅 [バイト/サイクル]	14	15

† 開発コード名



整数ループライン解析による
整数計算の最適化

整数ループライン解析による整数計算の最適化

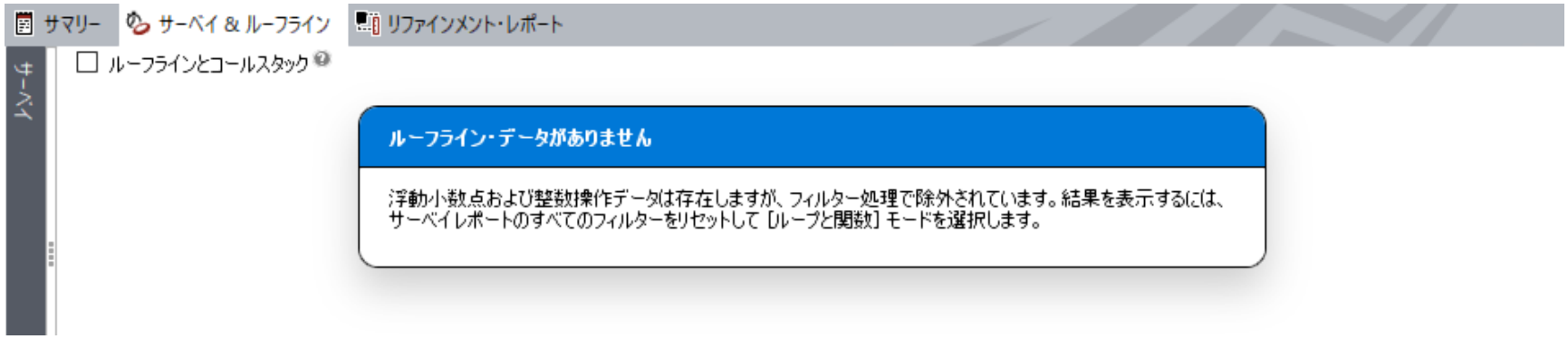
これまでのバージョンでは、浮動小数点データを扱うコードのみをループラインで解析

- FP + INT 混在の場合は FP データのみ表示
- INT データ型のループは非表示

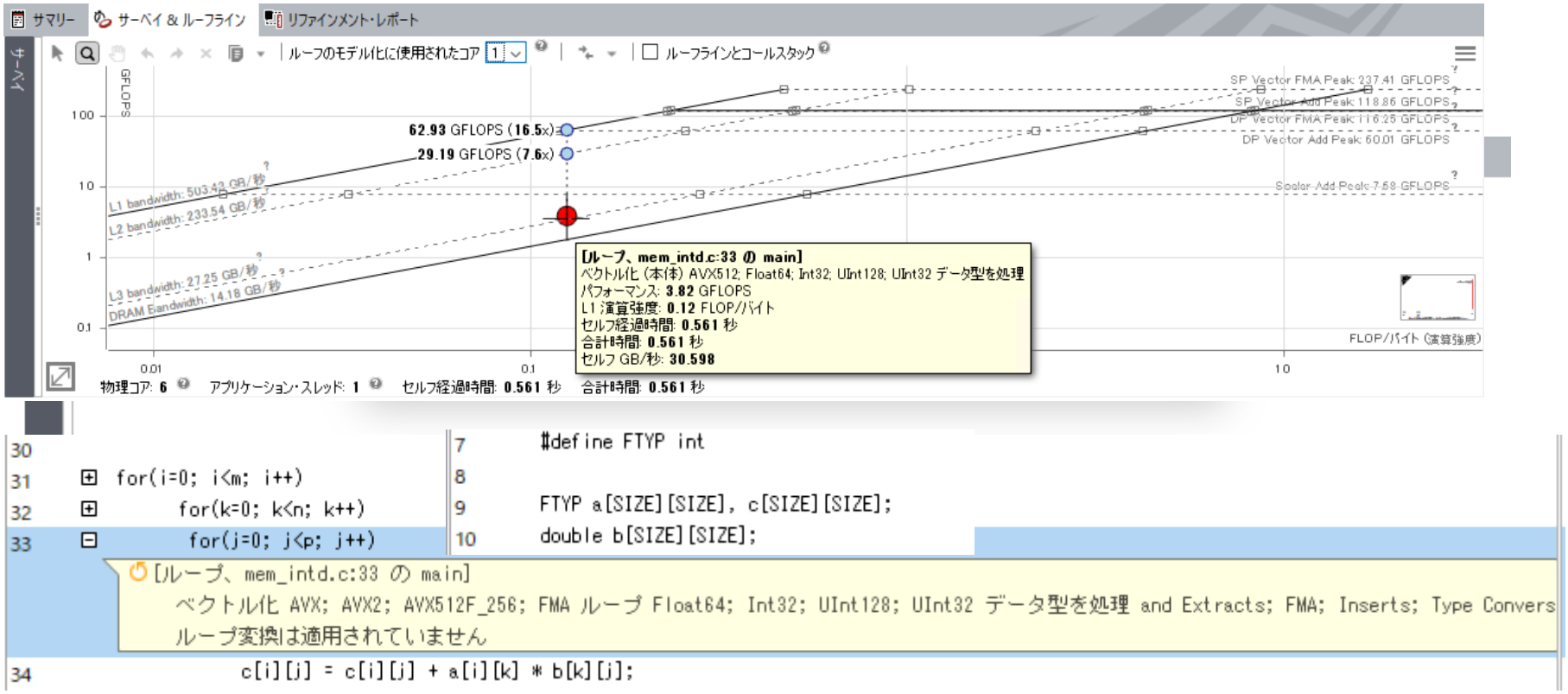
```
for(i=0; i<m; i++)
        for(k=0; k<n; k++)
            for(j=0; j<p; j++)
                c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

インテル® Advisor 2019 に加え、インテル® Advisor 2018 update4 でこの機能を利用できます

2018 update3 では整数データを表現できない



2018 update3 では整数データを表現できない

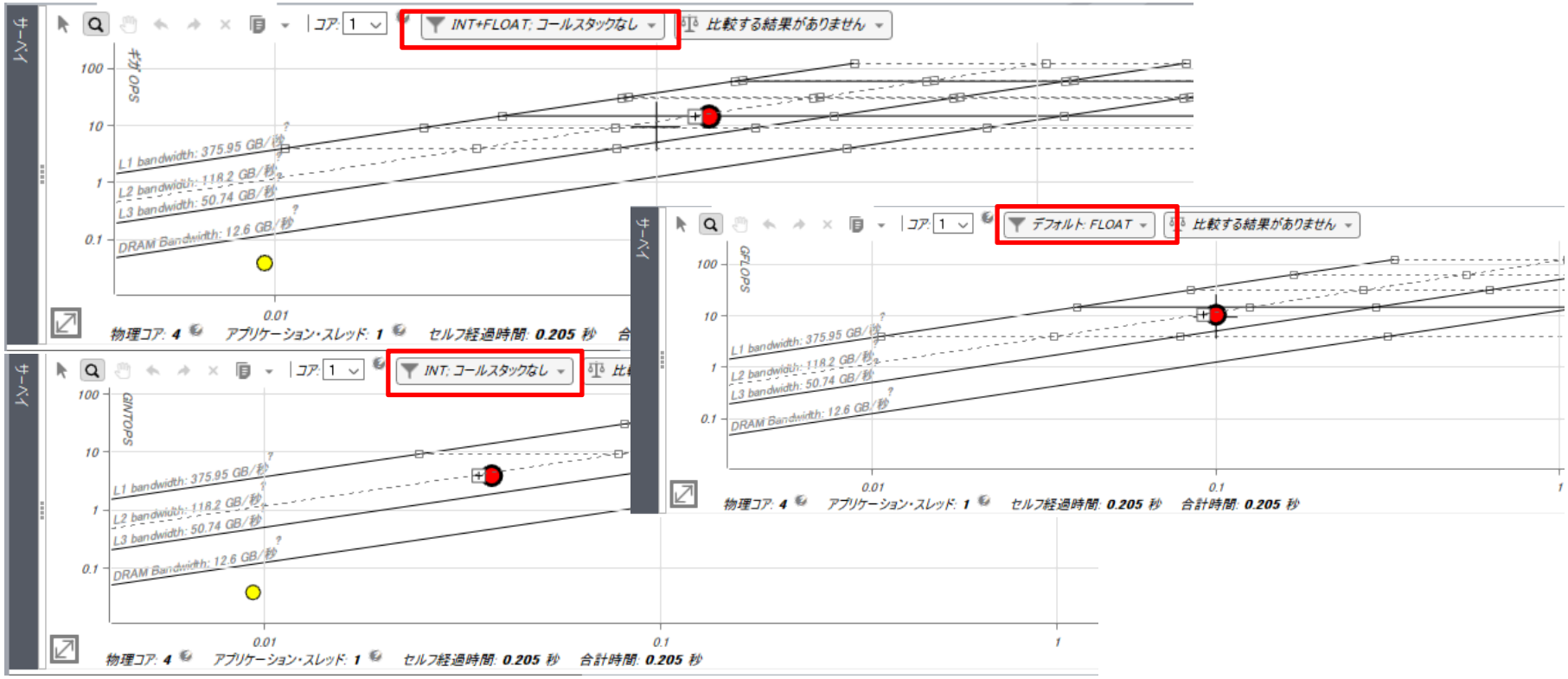



```
30
31      for(i=0; i<m; i++)
32          for(k=0; k<n; k++)
33              for(j=0; j<p; j++)
34                  c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

[ループ、mem_intd.c:33 の main]
ベクトル化 AVX; AVX2; AVX512F_256; FMA ループ Float64; Int32; UInt128; UInt32 データ型を処理 and Extracts; FMA; Inserts; Type Convers
ループ変換は適用されていません

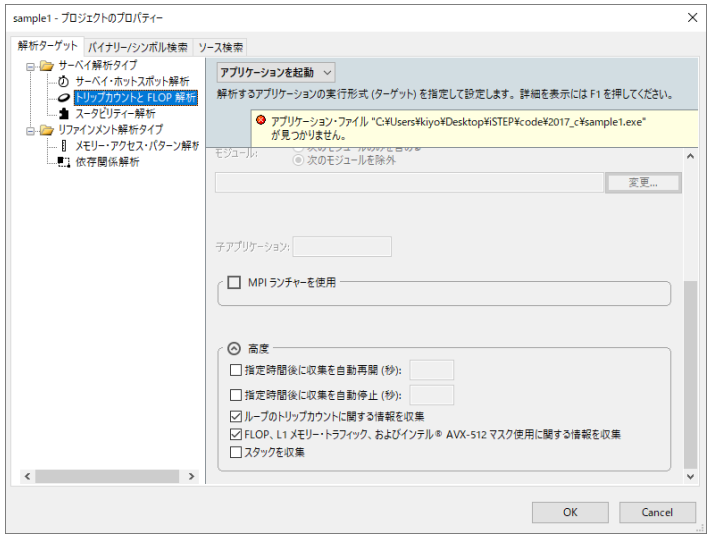
```
7       #define FTYP int
8
9       FTYP a[SIZE][SIZE], c[SIZE][SIZE];
10      double b[SIZE][SIZE];
```

Advisor 2019 のルーフライン表示切替



正確なメモリー使用量を取得し、キャッシュ・シミュレーションを使用して複数のハードウェア構成をチェック

ルーフラインとサーベイでのキャッシュ動作のモデル化

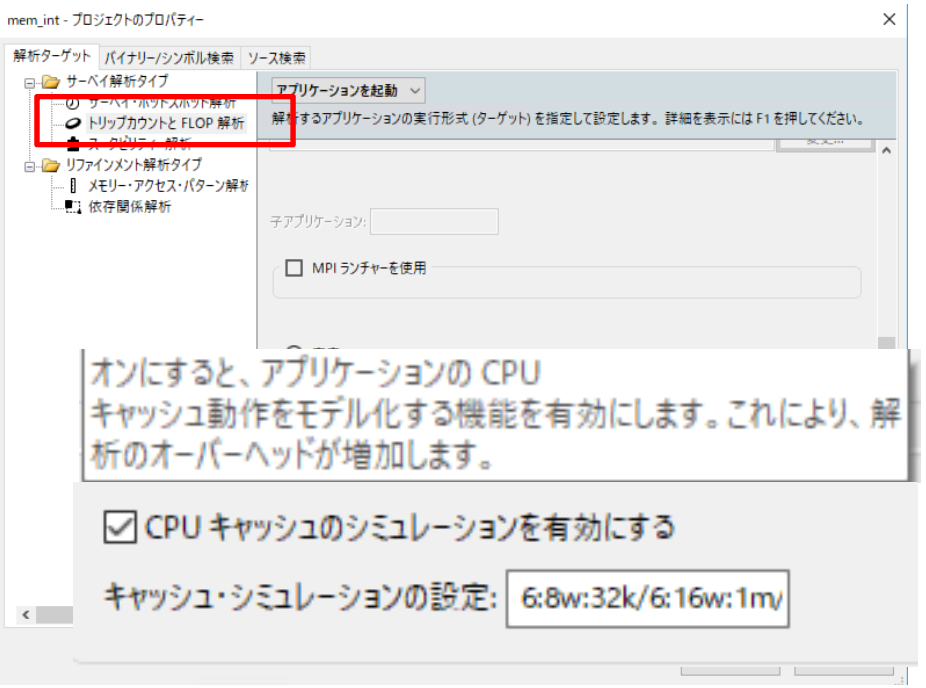


- キャッシュ･シミュレーション機能が拡張され、統合ルーフライン･ビューの機能が有効になります
- このバージョンのキャッシュ･シミュレーターは、各ループでロードおよびストアされたバイト数などのデータに対する複数レベルのキャッシュをモデル化します
- オプションで、特定のキャッシュ構成のシミュレーションを設定できます

この機能を使用するには、
環境変数 *ADVIXE_EXPERIMENTAL=int_roofline* を設定

キャッシュ動作のモデル化

環境変数 *ADVIXE_EXPERIMENTAL=int_roofline* を設定します



L1 キャッシュ
8 ウェイ
32K バイト
6 コア

L2 キャッシュ
16 ウェイ
1M バイト
6 コア

L3 キャッシュ
11 ウェイ
8.3M バイト

6:8w:32k/6:16w:1m/11w:8.3m

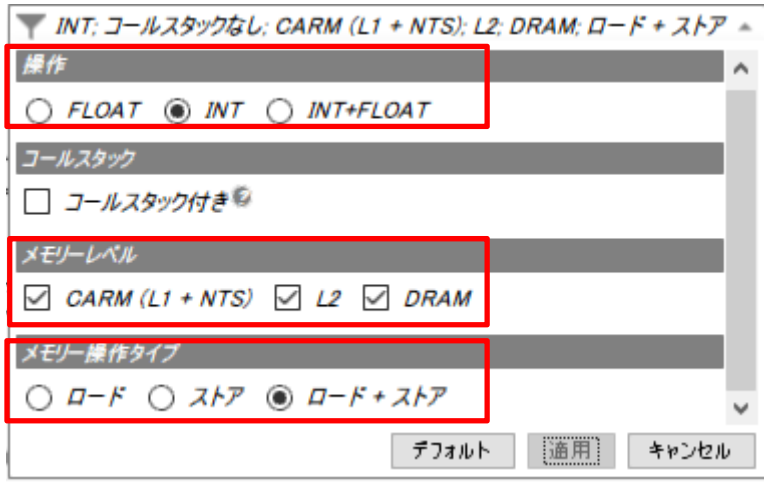
- 構成の形式は、レベル 1 から始まる「/」で区切ったそれぞれのキャッシュレベルを指定
- 各レベルは、「カウント:ウェイ:サイズ」の形式

ルーフラインでキャッシュのモデル化を設定

データ型

メモリーレベル

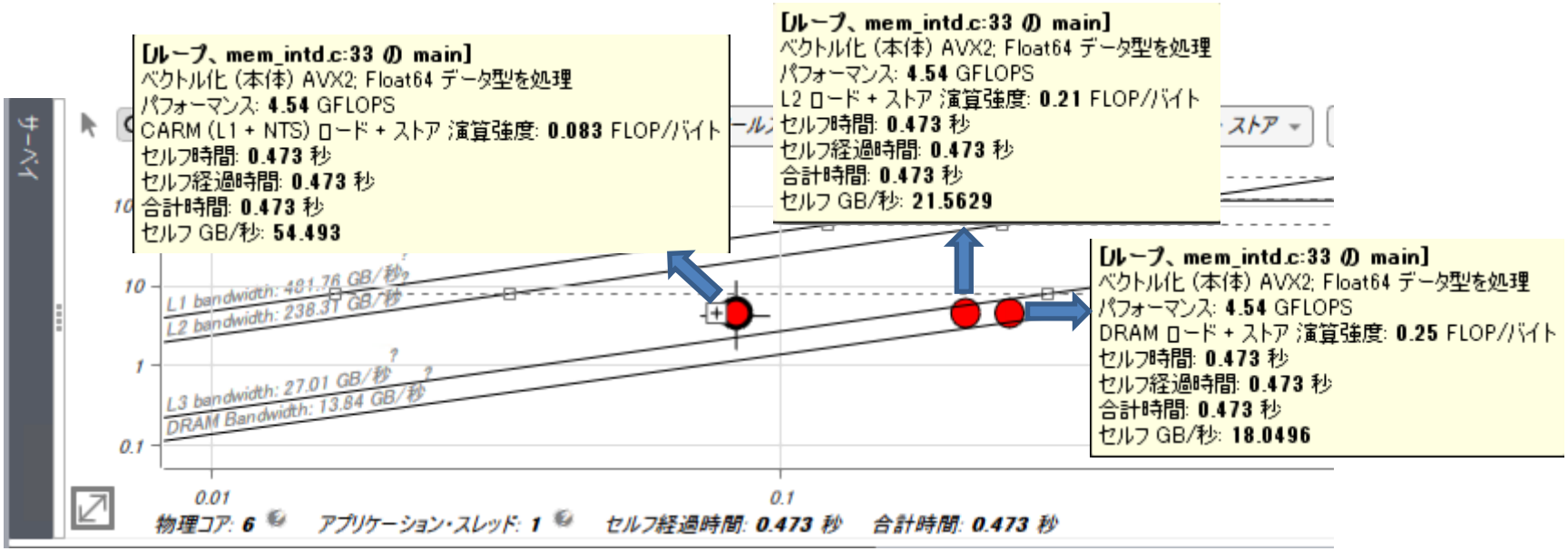
操作タイプ



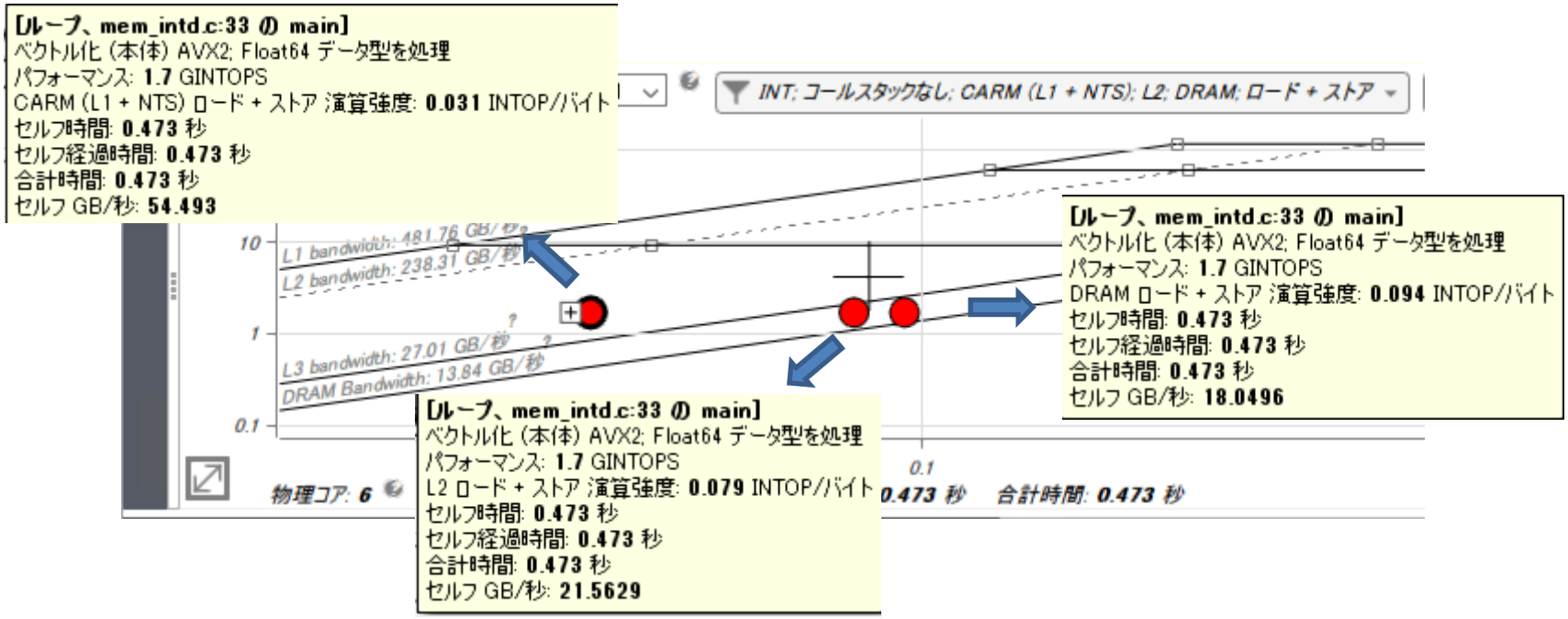
CARM: Cache Aware Roof model

キャッシュ･シミュレーションはデータ収集時に行われるため、
収集済みのデータを環境変数せずに起動した Advisor でも表示できます

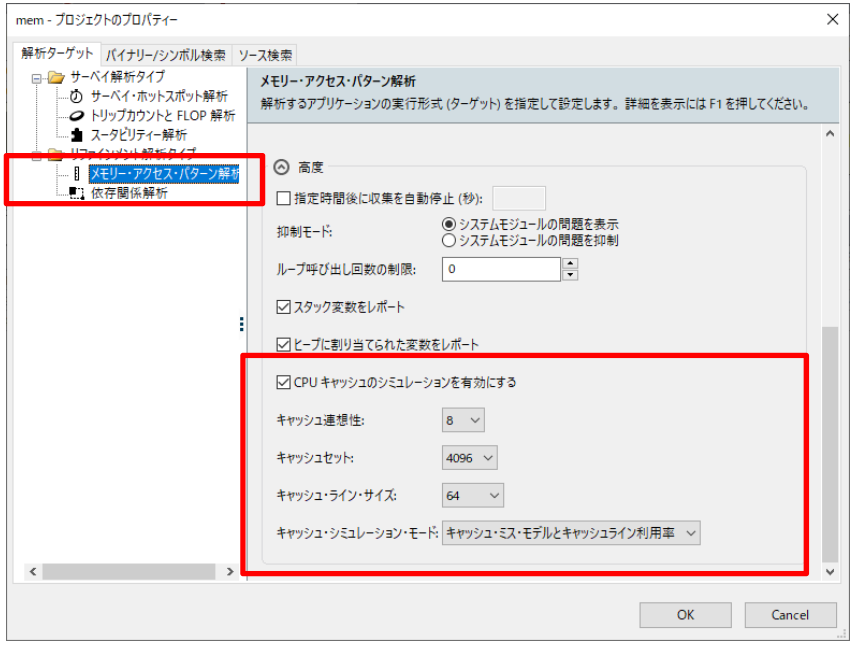
double 型のパフォーマンスのモデル



int 型のパフォーマンスのモデル



MAP 解析でシミュレーションを有効にする



- インテル® Advisor にキャッシュ・シミュレーター機能が追加されました
- これにより、正確なメモリー使用量と失われたアプリケーションの情報が得られます
- この機能を有効にして、[プロジェクトのプロパティー] の [メモリー・アクセス・パターン解析] でキャッシュの構成を設定できます
- シミュレーターは、モデルミスと、キャッシュラインの利用率または利用量のいずれかに設定できます

MAP 解析

パフォーマンス問題	キャッシュライン使用率	メモリーロード	メモリーストア	キャッシュミス	RFO キャッシュミス	ダーティー排出
	n/a	254	127	128	0	0
1 非効率なメモリー・アクセス・パターンがあります	n/a	744	124	622	0	0

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	シミュレートされたメモリー容量
[ループは main にあります: mem.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	4KB	8KB	0B
[ループは main にあります: mem.c ..	利用可能な情報がありません	0% / 100% / 0%	すべて定数ストライド	2MB	2MB	0B

```
39      start = clock();
40
41      for(i=0; i<m; i++)
42          for(k=0; k<n; k++)
43           for(j=0; j<p; j++)
```

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	命令ごとの最大アドレス範囲	モジュール	サイト名	アクセスタイプ
P1		4; 8192	一定ストライド	mem.c:44		a, b, c	2MB	mem.exe	loop_site_28	読み取り
P2		4	一定ストライド	mem.c:44		c	4KB	mem.exe	loop_site_28	書き込み
P4			並列サイト情報	mem.c:41				mem.exe	loop_site_28	

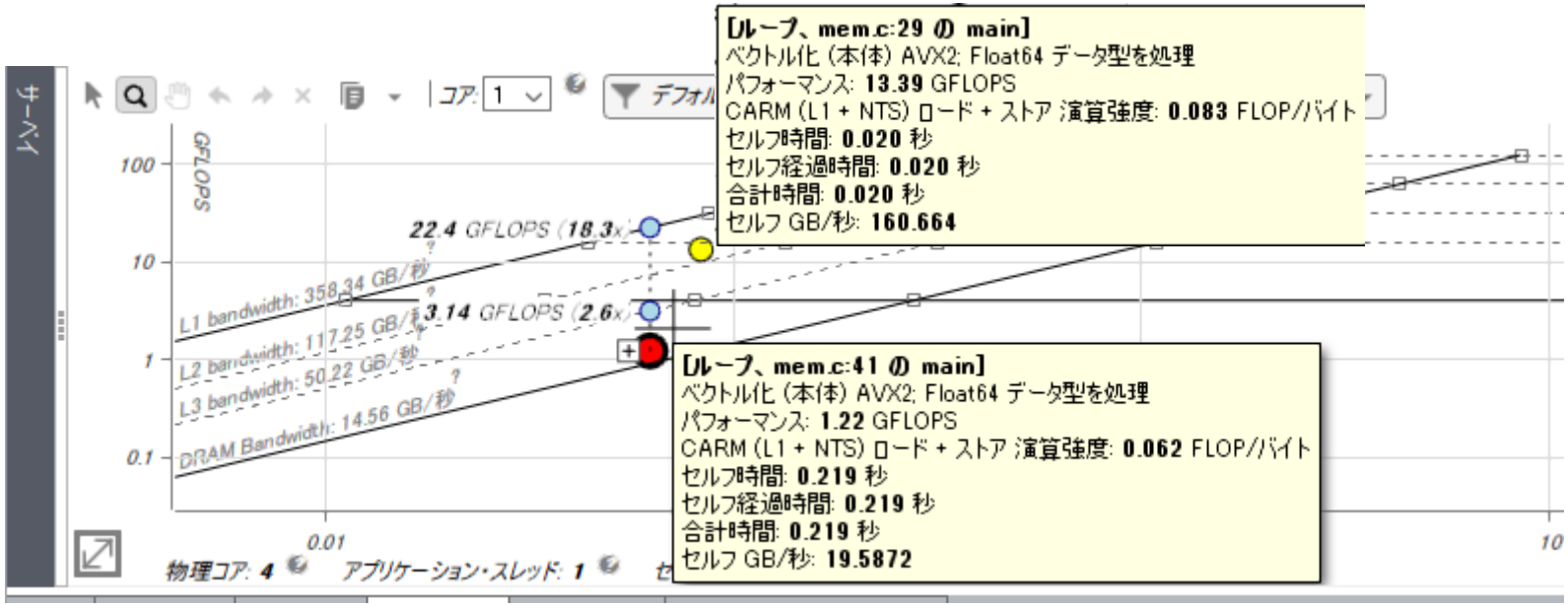
実際のアプリケーションで検証

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

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

メモリーのアクセスパターンが異なる: icl mem.c /Zi /QxCORE-AVX2 /O2

ルーフラインで確認



サーベイレポートを確認

関数呼び出しサイトとループ	パフォーマンス...	セルフ時間	合計時間	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				計算パフォーマンス			
						ベクト.	効率	予測.	VL (...	セルフ GFL ...	セルフ GFL ...	セルフ経過 ...	合計経過 ...
[ループ、mem.c:41 の main]	1 非効率なメ...	0.219s	0.219s	ベクトル化 (本体)		AVX2	64%	2.57x	4	1.224	0.268	0.219秒	0.219秒
[ループ、mem.c:29 の main]		0.020s	0.020s	ベクトル化 (本体)		AVX2	100%	4.57x	4	13.389	0.268	0.020秒	0.020秒

関数呼び出しサイトとループ	メモリー									
	セルフメモリー (GB)	セルフメモリー (GB/秒)	セルフロード GB	セルフストア GB	セルフ L2 ロード GB	セルフ L2 ストア GB	セルフ L3 ロード GB	セルフ L3 ストア GB	セルフ DRAM ロード GB	セルフ DRAM ストア GB
[ループ、mem.c:41 の main]	4.295	19.587	3.221	1.074	8.864	0.972	2.079	0.002	0	0
[ループ、mem.c:29 の main]	3.221	160.664	2.147	1.074	1.077	0.002	1.079	0.002	0.004	0

キャッシュ・シミュレーションが有効な場合にのみ表示

セルフ L2 ロード GB	セルフ L2 ストア GB	セルフ L3 ロード GB	セルフ L3 ストア GB
0	0	0	0
0	0	0	0

関数呼び出しサイトとループ	トリップカウント						命令セット解析			詳細
	平均	最小	最大	呼び出し数	反復期間	ループ合計時間	特性	データ型	ベクト.	
[ループ、mem.c:41 の main]	32	32	32	262144	< 0.001秒	< 0.001秒	FMA; Inserts	Float64	6	アンロール by 4; ...
[ループ、mem.c:29 の main]	128	128	128	262144	< 0.001秒	< 0.001秒	FMA	Float64		

キャッシュ･シミュレーションと
メモリー･アクセス･パターン解析で見るキャッシュのアーキテクチャーの影響

ベンチマーク･プログラム

```
for(j=1;j<n;j++){
  for (i=1,f=0;i<n; i++){
    f+=a[i]*b[i];
  }
  c[j]+=f;
}
```

```
n=40000
double a[n],b[n],c[n]
```

ブロッキング

```
FACTOR=4096,2048,1024,512,256,128,80,64
```

```
for(jj=1;jj<n/FACTOR+1;jj++){
  for(ii=1;ii<n/FACTOR+1;ii++){
    for(j=(jj-1)*FACTOR+1; j<min(jj*FACTOR,n);j++){
      for(i=(ii-1)*FACTOR+1,f=0 ;i<min(ii*FACTOR,n);i++){
        f+=a[i]*b[i];
      }
      c[j]+=f;
}}}
```

ベンチマーク：同一プログラムを実行

ファクター	i7-7800X	i7-5820K	i7-4790K
4096	0.248 / 0.264	0.652 / 0.366	0.481 / 0.251
2048	0.248 / 0.254	0.632 / 0.283	0.481 / 0.236
1024	0.246 / 0.299	0.618 / 0.324	0.482 / 0.236
512	0.248 / 0.396	0.622 / 0.410	0.483 / 0.265
256	0.246 / 0.642	0.623 / 0.625	0.483 / 0.517
128	0.248 / 1.288	0.630 / 1.216	0.482 / 1.010
80	0.248 / 2.360	0.620 / 2.152	0.480 / 1.791
64	0.247 / 3.230	0.624 / 2.866	0.482 / 2.373

ベンチマーク：同一プログラムを実行

ファクター	i7-7800X	i7-5820K	i7-4790K
4096	0.248 / 0.264	0.652 / 0.366	0.481 / 0.251
2048	0.248 / 0.254	0.632 / 0.283	0.481 / 0.236
1024	0.246 / 0.299	0.618 / 0.324	0.482 / 0.236
512	0.248 / 0.396	0.622 / 0.410	0.483 / 0.265
256	0.246 / 0.642	0.623 / 0.625	0.483 / 0.517

Cache

L1 Data	6 x 32 KBytes	8-way
L1 Inst.	6 x 32 KBytes	8-way
Level 2	6 x 1 MBytes	16-way
Level 3	8.25 MBytes	11-way

Cache

L1 Data	6 x 32 KBytes	8-way
L1 Inst.	6 x 32 KBytes	8-way
Level 2	6 x 256 KBytes	8-way
Level 3	15 MBytes	20-way

Cache

L1 Data	4 x 32 KBytes	8-way
L1 Inst.	4 x 32 KBytes	8-way
Level 2	4 x 256 KBytes	8-way
Level 3	8 MBytes	16-way

i7-7800X vs. i7-5820K

プログラムのメトリック

経過時間	0.52秒		
ベクトル命令セット	AVX2, AVX, SSE	CPU スレッド数	1
合計 GFLOP 数	6.45	合計 GFLOPS	12.31
合計演算強度	0.12305		

ループのメトリック

メトリック	合計	
合計 CPU 時間	0.48秒	100.0%
2 でベクトル化されたループ s の時間	0.45秒	93.5%
スカラーコード時間	0.03秒	6.5%
合計 GFLOP 数	6.45	100.0%
合計 GFLOPS	12.31	

ベクトル化のゲイン/効率

ベクトルループのゲイン/効率	4.25x	53%
プログラムのおよそのゲイン	4.04x	

プログラムごとの推奨

⚠ 上位の命令セット・アーキテクチャー (ISA) を利用できます

より上位の ISA を使用してアプリケーションを再コンパイルすることを検討してください。詳細を表示

時間を消費している上位のループ

ループ	セルフ時間	合計時間	トリップカウント
[block.c:49 の main にあるループ]	0.234秒	0.234秒	127; 3
[block.c:34 の main にあるループ]	0.217秒	0.217秒	2499; 3
[block.c:48 の main にあるループ]	0.016秒	0.250秒	2047
[block.c:47 の main にあるループ]	0.016秒	0.266秒	1248
[block.c:32 の main にあるループ]	0秒	0.217秒	39999

収集の詳細

プラットフォーム情報

CPU 名	Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz
周波数	3.50 GHz
論理 CPU 数	12

プログラムのメトリック

経過時間	1.74秒		
ベクトル命令セット	AVX2, AVX, SSE	CPU スレッド数	1
合計 GFLOP 数	6.45	合計 GFLOPS	3.70
合計演算強度	0.12305		

ループのメトリック

メトリック	合計	
合計 CPU 時間	1.68秒	100.0%
2 でベクトル化されたループ s の時間	1.65秒	98.1%
スカラーコード時間	0.03秒	
合計 GFLOP 数	6.45	100.0%
合計 GFLOPS	3.70	

ベクトル化のゲイン/効率

ベクトルループのゲイン/効率	4.10x	51%
プログラムのおよそのゲイン	4.04x	

時間を消費している上位のループ

ループ	セルフ時間	合計時間	トリップカウント
[block.c:34 の main にあるループ]	1.321秒	1.321秒	2499; 3
[block.c:49 の main にあるループ]	0.328秒	0.328秒	127; 3
[block.c:48 の main にあるループ]	0.031秒	0.359秒	2047
[block.c:32 の main にあるループ]	0秒	1.321秒	39999
[block.c:47 の main にあるループ]	0秒	0.359秒	1248

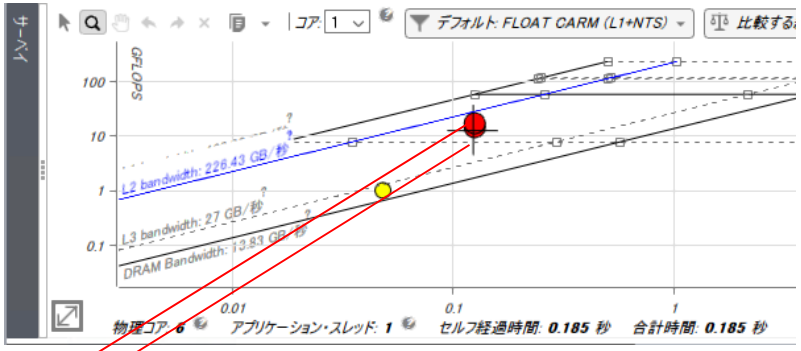
収集の詳細

プラットフォーム情報

CPU 名	Intel(R) Core(TM) i7-5820K CPU @ 3.30GHz
周波数	3.30 GHz
論理 CPU 数	12

i7-7800X

FACTOR=2048 の場合



GB/秒

関数呼び出しサイトとループ	メモリー										トリップカウント	
	セルフメモリー (GB)	セルフメモリ...	セルフロード GB	セルフストア GB	セルフ L2 ロード GB	セルフ L2 ストア GB	セルフ L3 ロード GB	セルフ L3 ストア GB	セルフ DRAM ロード GB	セルフ DRAM ストア GB	平均	呼び出し数
[ループ、block.c:34 の main]	25.597	117.035	25.597	0	25.563	0.003	< 0.001	< 0.001	0	0	2499; 3	39999; 39
[ループ、block.c:49 の main]	25.536	137.921	25.536	0	14.416	0.028	< 0.001	< 0.001	0	0	127; 3	799600; 79
[ループ、block.c:48 の main]	1.290	21.210	0.875	0.415	0.314	0.095	< 0.001	< 0.001	0	0	2047	24960

MAP 解析

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	シミュレートされたメモリー容量	サイト名	パフォーマンス問題	キャッシュライン使用率	メモリーロード	メモリーストア	キャッシュミス
ループは main にあります: block.c ...	利用可能な情報がありません	0% / 100% / 0%	すべて定数ストライド	312KB	625KB	0B	loop_site_1	1 非効率なメモリ...	n/a	19984	0	9994
ループは main にあります: block.c ...	利用可能な情報がありません	0% / 100% / 0%	すべて定数ストライド	16KB	32KB	0B	loop_site_3	1 非効率なメモリ...	n/a	1008	0	506

```
for(j=1;j<n;j++){
#pragma omp parallel for reduction(+:f)
    for (i=1, f=0;i<n; i++){
        f+=a[i]*b[i];
    }
```

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	命令ごとの最大アドレス範囲	モジュール	サイト名	アクセスタイプ
P1	1; 4	一定ストライド	block.c:35		a, b	312KB			

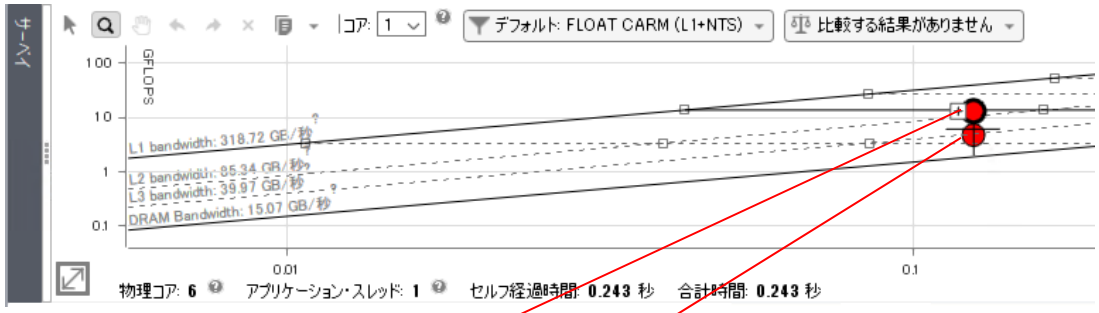
```
33  #pragma omp parallel for reduction(+:f)
34      for (i=1, f=0;i<n; i++){
35          f+=a[i]*b[i];
36      }
37      c[j]+=f;
```

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	命令ごとの最大アドレス範囲	モジュール	サイト名	アクセスタイプ
P2	1; 4	一定ストライド	block.c:50		a, b	16KB	block9.exe	loop_site_3	読み取り
P4		並列サイト情報	block.c:49				block9.exe	loop_site_3	

i7-5820K

FACTOR=2048 の場合



GB/秒

関数呼び出しサイトとループ	メモリー										トリップカウント	
	セルフメモリー (GB)	セルフメモリ...	セルフロード GB	セルフストア GB	セルフ L2 ロード GB	セルフ L2 ストア GB	セルフ L3 ロード GB	セルフ L3 ストア GB	セルフ DRAM ロード GB	セルフ DRAM ストア GB	平均	呼び出し数
[ループ、block.c:34 の main]	25.589	38.010	25.589	0	25.535		25.535	0	< 0.001	0	2499	39999
[ループ、block.c:49 の main]	25.383	104.668	25.383	0	13.395		0.013	0	0	0	127	799600

MAP 解析

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			サイト名	パフォーマンス問題	キャッシュライン使用率	メモリーロード	メモリーストア	キャッシュ
				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	シミュレートされたメモリー容量						
[ループは main にあります: block.c ...	利用可能な情報がありません	0% / 100% / 0%	すべて定数ストライ	312KB	625KB	0B	loop_site_1	1 非効率なメモリー・アクセス・パターンがあります	n/a	19984	0	9994
[ループは main にあります: block.c ...	利用可能な情報がありません	0% / 100% / 0%	すべて定数ストライ	16KB	32KB	0B	loop_site_3	1 非効率なメモリー・アクセス・パターンがあります	n/a	1008	0	506

さらに詳細な解析を行うには、インテル® VTune™ Amplifier 2019 を合わせて利用します

インテル® VTune™ Amplifier 2019 では、使い勝手が大幅に改善され、より目的に適した操作と解析が可能となりました

インテル® Advisor 2019 の日本語版と共に、インテル® VTune™ Amplifier 2019 の日本語版を開発中です

これら以外の拡張機能

ルーフラインの拡張

- コールスタック付きのルーフラインが利用可能になりました
- ルーフラインの動的 HTML コピーのエクスポートによって結果を共有:
  advixe-cl –report roofline –report-output=path/to/output.html
- 複雑なグラフの解析を簡素化し、選択したドットをビットマップとして保存するため、ルーフライン・グラフ上の任意のドットのサブセットをフィルター処理できます
- アプリケーションの実用パフォーマンスの限界を見るため、カスタムスレッド数にルーフラインを調整する機能
- ループラインのベンチマークが複数ランクの MPI アプリケーションで同期されるようになったため、同じ物理ノードで実行されるすべてのランクでルーフの値が同じになります
- 同じグラフで複数のルーフライン結果を比較する機能
- 単一の CLI コマンドでルーフライン・データを収集: advixe-cl –collect roofline

選択可能なプロファイルを使用して高速に解析結果を得るためオーバーヘッドを軽減

- プロジェクトのプロパティーでループ呼び出しカウントを制限し、解析時間を設定することで、メモリー・アクセス・パターンと依存関係解析のオーバーヘッドを軽減
- 解析範囲を狭めてオーバーヘッドを軽減するため、ルーフライン、FLOPS とトリップカウント収集でプロファイルを選択する機能を追加

ユーザビリティーの向上

- オプションメニューでフォントサイズをカスタマイズできるようになりました。これは、SSH の X 転送セッションで GUI の表示を調整するのに役立ちます
- コマンドラインから、ソースファイルと行レベルでループを選択する機能:
  advixe-cl –mark-up-loops –select main.cpp:12,other.cpp:198
- Pythoin* を使用して簡単に HTML レポートを作成:
  advixe-python to_html.py ProjectDir

推奨事項タブの改善:

- 推奨事項タブを一新し使いやすいレイアウトに
- 推奨される特定のアンロール係数値や、インライン展開される関数名など、ユーザーコード固有の推奨事項のパラメーター
- 新たな推奨事項: メモリー依存のアプリケーションのパフォーマンスを改善する、非テンポラルストア (NTS) 命令の使用

iSUS