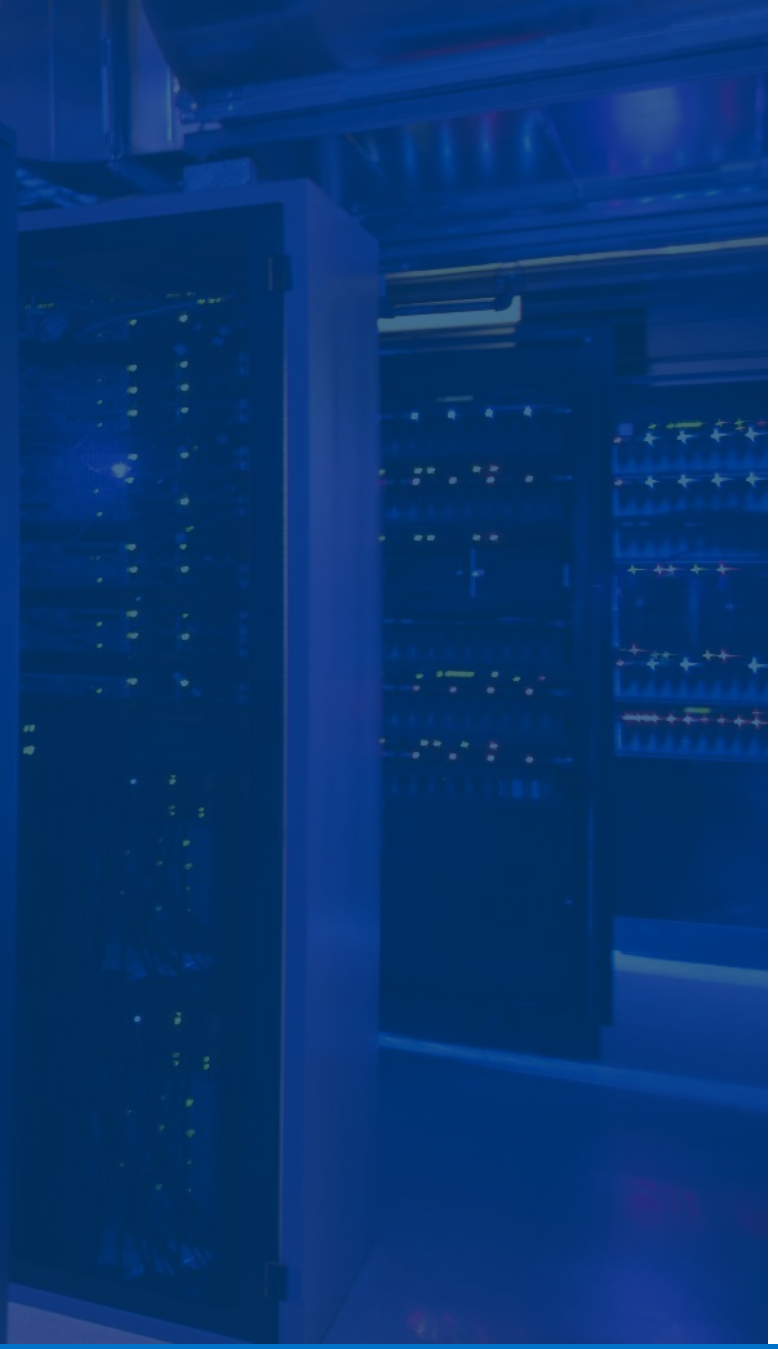
インテル® Advisor の GPU 向け機能の概要



内容

1 インテル® Advisor による CPU ルーフライン

2 インテル® Advisor によるベクトル化

3 インテル® Advisor によるオフロードのモデル化

4 インテル® Advisor による GPU ルーフライン

GPU 計算向けのインテル解析ツール

インテル® Advisor

オフロードのモデル化 (オフロード・アドバイザー)

- GPU オフロードにより大きな効果が得られる候補の特定
- ボトルネックと主な制限要因を検出
- パフォーマンス、ヘッドルーム、ボトルネックをモデル化することで、ハードウェアを入手する前にコードを準備することができます

ルーフライン解析

- ハードウェアの上限に対するパフォーマンスのヘッドルームを確認
- ボトルネックとどの最適化が最も効果的であるかを特定し、パフォーマンス最適化の方針を決定します
- 最適化の状況を可視化

フローグラフ・アナライザー

- CPU/GPU コードを可視化し、CPU デバイスの推奨値を取得

インテル® VTune™ プロファイラー

オフロードのパフォーマンス・チューニング

- プラットフォーム上の各種 CPU コアや GPU コアでのコード実行を調査できます
- CPU と GPU のアクティビティーを関連付けます
- アプリケーションが GPU 依存であるか CPU 依存であるかを特定します

HPC パフォーマンス特性

- OpenMP* アプリケーションがワークを効率良く GPU にオフロードしているか確認します

GPU 計算/メディア・ホットスポット

- 最も時間を要する GPU カーネルを解析し、GPU ハードウェア・メトリックに基づいて GPU の使用率を特徴付けます
- ソース行およびカーネル・アセンブリー・レベルでの GPU コードのパフォーマンスを表示します

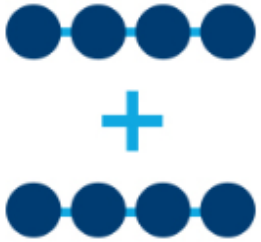
ハイパフォーマンスなコードを設定するための豊富な機能セット
インテル® Advisor

オフロードのモデル化 (オフロード・アドバイザー)

GPU 向けのオフロード戦略と
モデルのパフォーマンスを設定

ルーフライン解析

アプリケーションの計算と
メモリーを最適化



ベクトル化の最適化

ベクトル化の並列性により
効率を改善

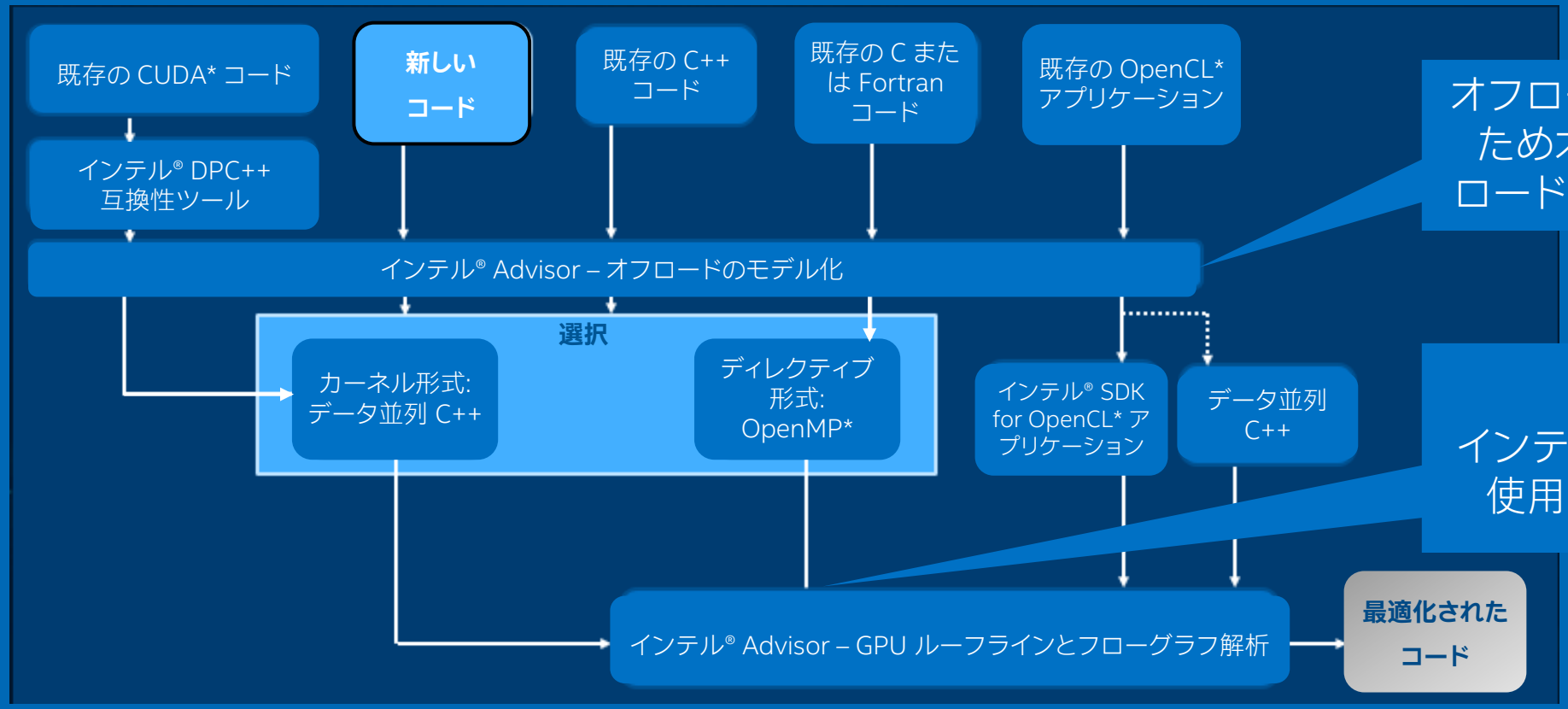
スレッドの プロトタイプ作成

マルチスレッド設計のモデル化、
チューニング、およびテスト

ヘテロジニアス アルゴリズムを設計

データフローと依存関係の
計算グラフを作成して解析

パフォーマンス向上のためインテル® Advisor を使用



オフロードするカーネルを特定するためオフロードのモデル化 (オフロード・アドバイザー) を使用します

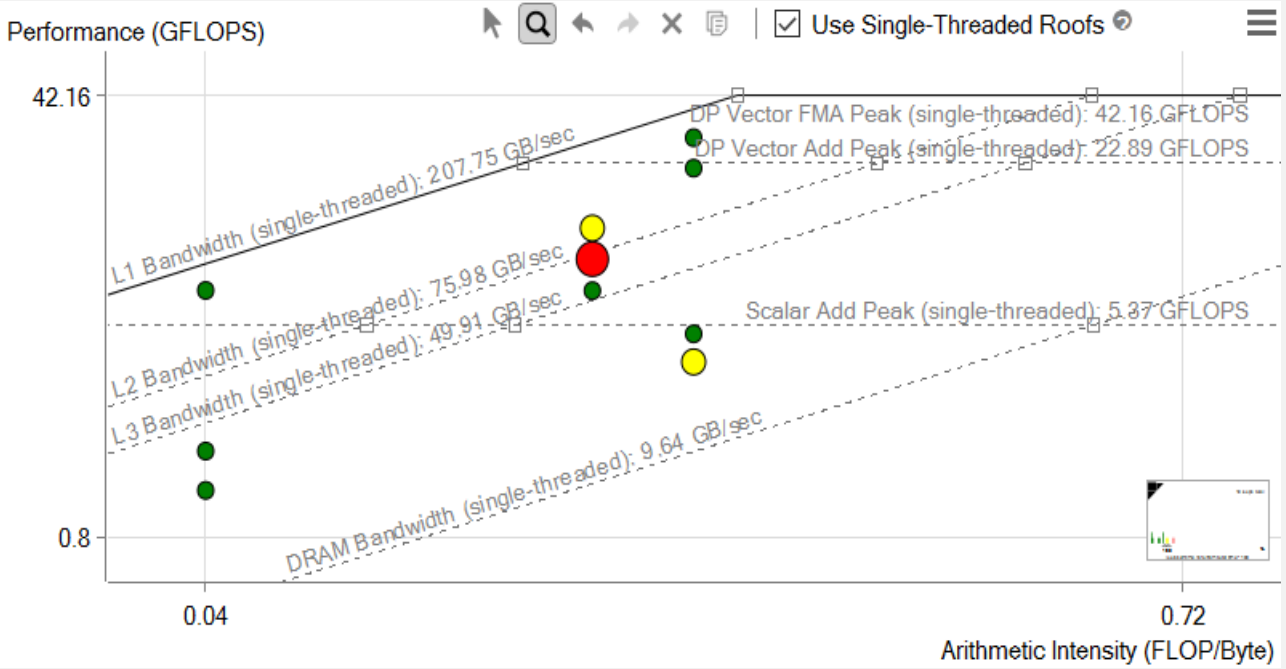
インテル® Advisor と インテル® VTune™ プロファイラーを使用してカーネルを最適化します

インテル® Advisor による CPU ルーフライン

ルーフライン・グラフとは?

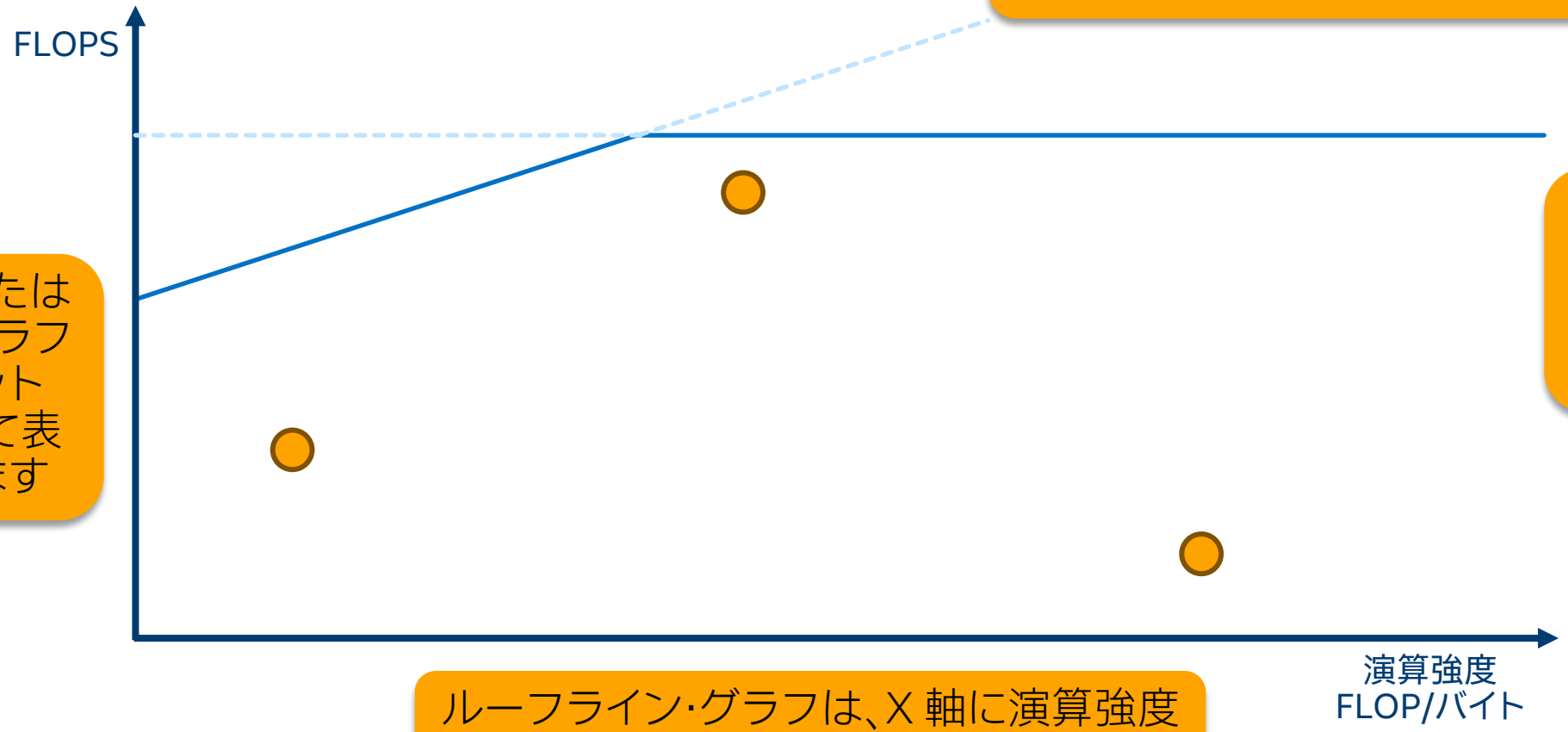
- ボトルネックの場所は?
- パフォーマンス改善の余地は?
- 対応できるボトルネックと対処すべきボトルネック
- 最も可能性の高い原因は?
- 次にすべきことは?

ハードウェアの性能上限に対する
アプリケーションのパフォーマンスを比較



ルーフライン・グラフを表示

操作 (ops)/バイト (算術強度) と 1 秒間に供給される最大バイトの積である最大 FLOPS が対角線で示されます



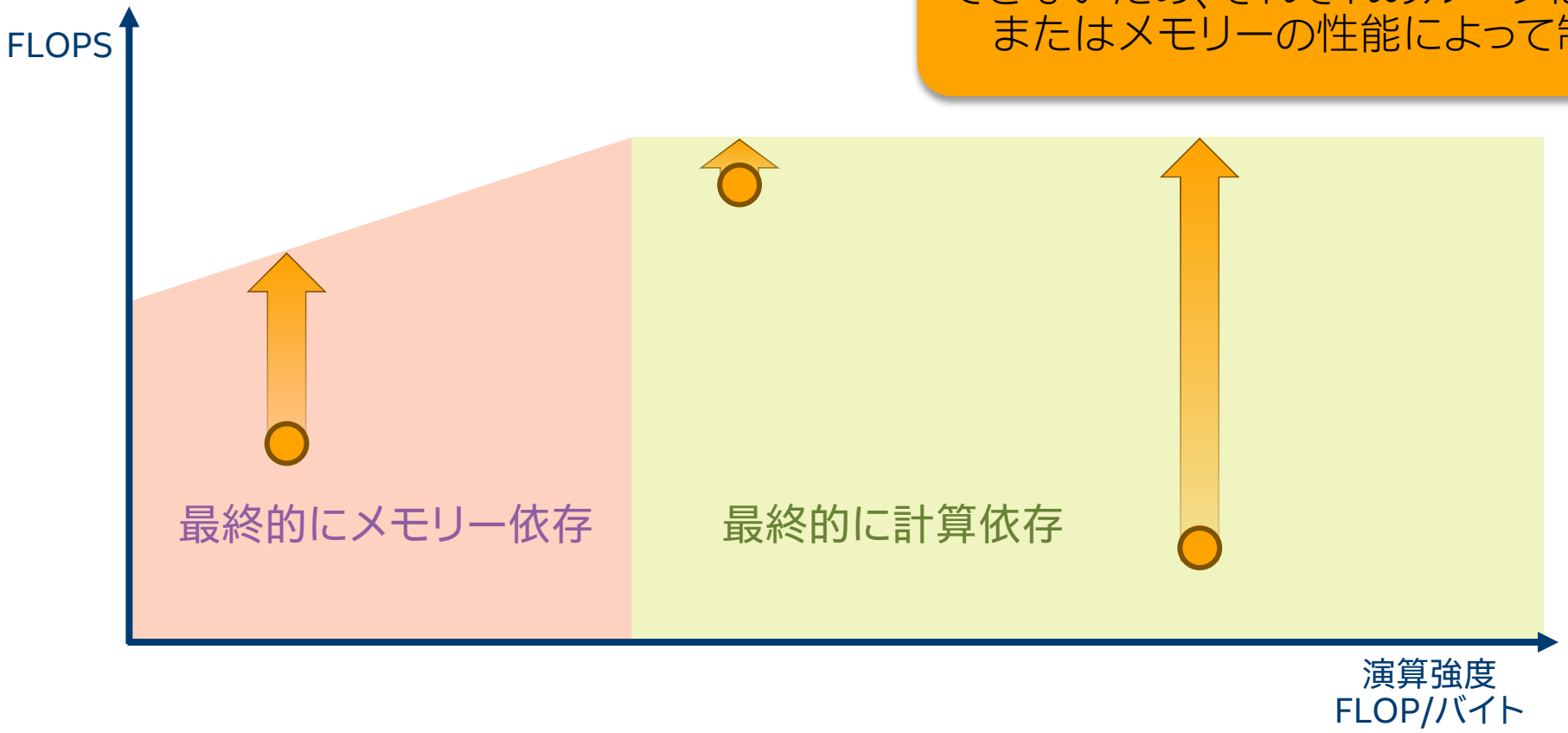
ループまたは関数はグラフ上にドット (点) として表示されます

CPU の最大 FLOPS は水平線で表示されます

ルーフライン・グラフは、X 軸に演算強度 (AI) を Y 軸に FLOPS を示します

最終的にパフォーマンスを制限するものは

パフォーマンスはマシンの能力を超えることはできないため、それぞれのループは最終的に計算またはメモリーの性能によって制限されます

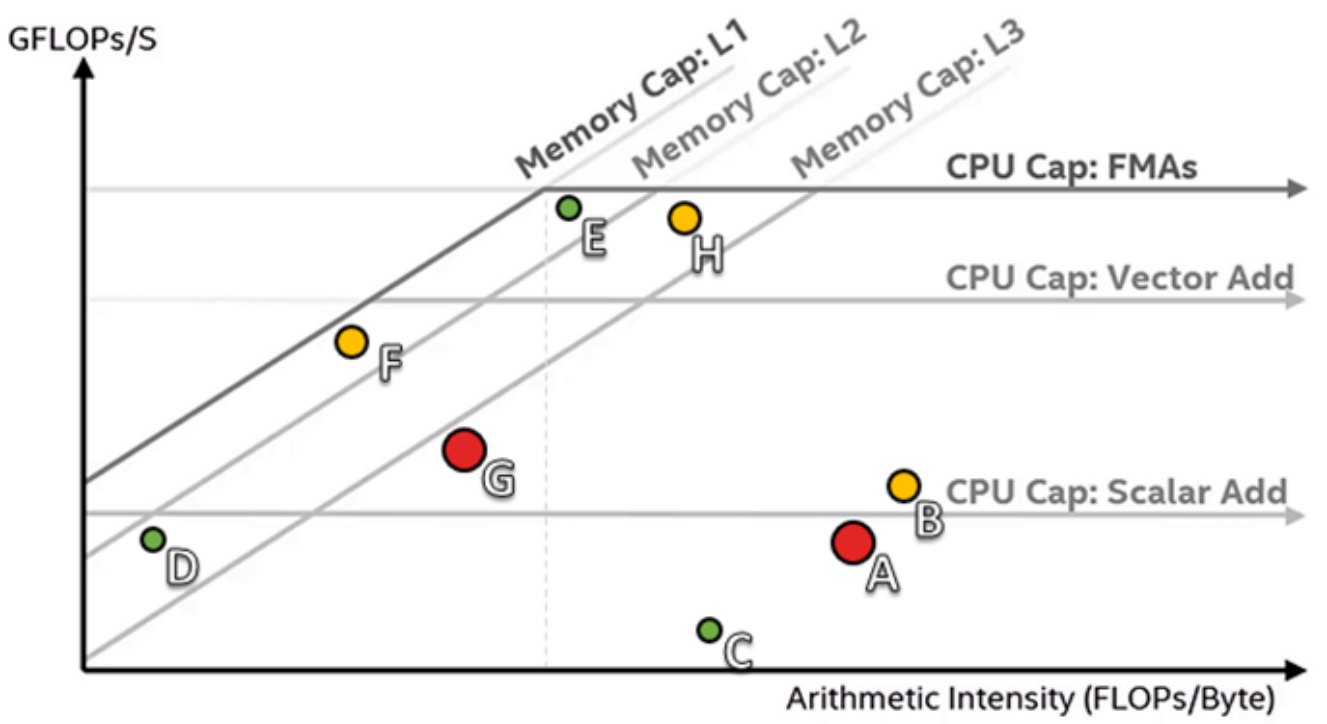


最良の最適化候補を特定

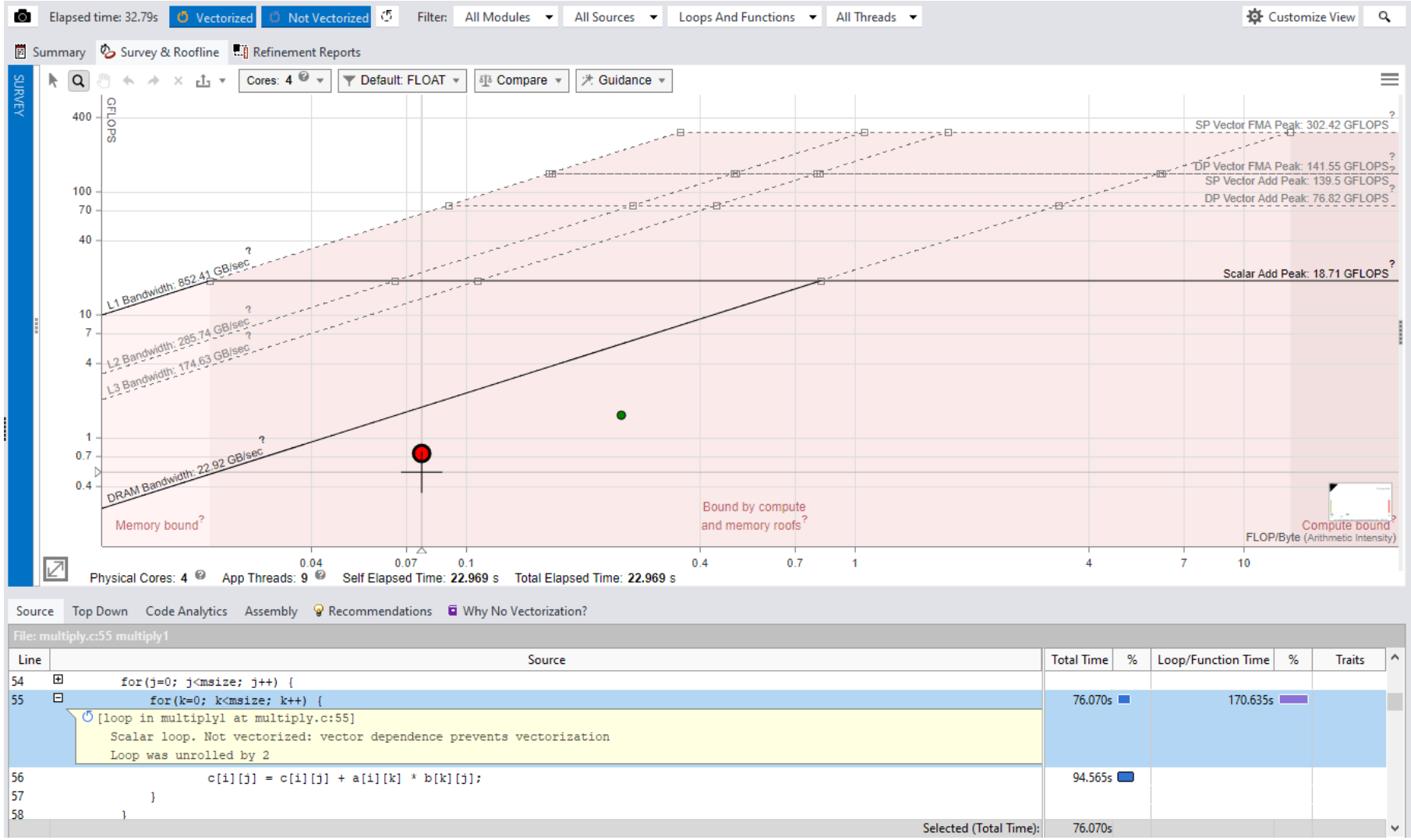
最適化の労力を最も効果的な場所に集中させます

- 大きくそして赤く表示されるループが最もインパクトがあります
- ルーフから離れているループには改善の余地があります

特定の計算タイプやキャッシュレベルに応じて、追加のループ情報を表示できます



CPU ルーフライン・グラフの例



インテル® Advisor による ベクトル化

データ領域の正しいデータ

ベクトル化に必要なすべてのデータを取り込む

ベクトル化されたループのフィルター処理

ベクトル化を妨げる要因

Elapsed time: 32.79s | Vectorized | Not Vectorized | Filter: All Modules | All Sources | Loops And Functions | All Threads

Summary | Survey & Roofline | Refinement Reports

ROOFLINE

Function Call Sites and Loops	Performance Issues	CPU Time: Total Time	CPU Time: Self Time	Type	Why No Vectorization?	Vectorized Loops: Vect...	Vectorized Loops: Efficiency	Vectorized Loops: Gain...
[loop in multiply1 at multiply.c:55]	**2 Assumed dependency present**	**170.635s**	**170.635s**	**Scalar**	**vector dependence prevents vectorization**			
[loop in multiply1 at multiply.c:54]		170.684s	0.049s	Scalar	outer loop was not auto-vectorized: consider ...			
[loop in init_arr at matrix.c:50]	2 Potential underutilization of FMA ...	0.011s	0.011s	Vectorized (Bo...		SSE2	100%	3.30x
ThreadFunction		170.684s	0.000s	Function				
multiply1		170.684s	0.000s	Function				
[loop in multiply1 ... multiply.c:53]		170.684s	0.000s	Scalar	outer loop was not auto-vectorized: ... ider ...			
_scrt_common_m... seh		0.085s	0.000s	Function				
main		0.085s	0.000s	Function				
init_arr	1 Data type conversions present	0.011s	0.000s	Function				
[loop in init_arr ... rix.c:49]	1 Data type conversions present	0.011s	0.000s	Scalar	inner loop was ... torized			
[loop in Paralle... y at thrmodel.	2 System function call(s) present	0.008s	0.000s	Scalar	loop cont... was found, but loop iter ...			
ParallelMultip...		0.065s	0.000s	Function				

ホットなループに注目

ベクトル化の問題は?

ベクトル化の命令セットは?

コードの効率は?

効率良いベクトル化のための 5 つのステップ

1. コンパイラーの診断 + パフォーマンス・データ + SIMD 効率情報
2. ガイド: 問題を特定して解決方法を推奨
3. “正確な” トリップカウント + FLOPS とマスク: 利用率、並列性の粒度、オーバーヘッドを理解
4. ループ伝搬依存の解析
5. メモリー・アクセス・パターンの解析

コードを効率良く ベクトル化

1. コンパイラーの診断 + パフォーマンス・データ + SIMD 効率情報

2. ガイド: 問題を特定して解決方法を推奨

クリックして推奨事項を表示

インテル® Advisor は、ループ反復をベクトル本体へ移動するヒントを示します

Elapsed time: 34.42s | Vectorized | Not Vectorized | Filter: All Modules | All Sources | Loops And Functions | All Threads | Customize View

Summary | Survey & Roofline | Refinement Reports

Function Call Sites and Loops	Performance Issues	CPU Time: Total Time	CPU Time: Self Time	Type	Why No Vectorization?	Vectorized Loops: Vect...	Vectorized Loops: Efficiency	Vectorized Loops: Gain...	Vectorized Loops: VL (...	Compute Performance: Self GFLOPS	Compute Performance: Self AI
[loop in multiply1 at multiply.c:55]	2 Assumed ...	172.445s	172.445s	Scalar	vector dependenc ...						
[loop in multiply1 at multiply.c:54]		172.490s	0.045s	Scalar	outer loop was not ...						
[loop in init_arr at matrix.c:80]	Potential u ...	0.019s	0.019s	Vectorized (Bo...		SSE2	~100%	3.30x	2	0.896	0.250
ThreadFunction		172.490s	0.000s	Function							0
multiply1		172.490s	0.000s	Function							
[loop in multiply1 at multiply.c:53]		172.490s	0.000s	Scalar	outer loop was not ...						

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

File: multiply.c:55 multiply1

Line	Source	Total Time	%	Loop/Function Time	%	Traits
54	for ... msize; j++) {					
55	... k=0; k<msize; k++) {	71.913s		172.445s		
	... multiply1 at multiply.c:55]					
	... ar loop. Not vectorized: vector dependence prevents vectorization					
	... oop was unrolled by 2					
	c[i][j] = c[i][j] + a[i][k] * b[k][j];	100.532s				
	Selected (Total Time):	71.913s				

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

All Advisor-detectable issues: C++ | Fortran

Assumed dependency present

The compiler assumed there is an anti-dependency (Write after read - WAR) or a true dependency (Read after write - RAW) in the loop. Improve performance by investigating the assumption and handling accordingly.

Confirm dependency is real

Run the Dependencies analysis to identify real data dependencies.

There is no confirmation that a real (proven) dependency is present in the loop.

Potential underutilization of FMA instructions

ループが実行された回数と
トリップカウントを特定します

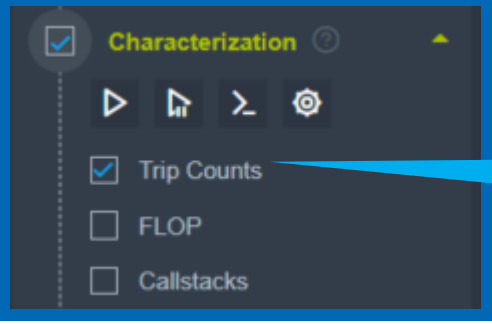
3. “正確な” トリップカウント + FLOPS とマスク: 利用率、並列性の粒度、オーバーヘッドを理解

Elapsed time: 32.79s | Vectorized | Not Vectorized | Filter: All Modules | All Sources | Loops And Functions | All Threads

Summary | Survey & Roofline | Refinement Reports

ROOFLINE

Function Call Sites and Loops	Performance Issues	CPU Time				Trip Counts	
		Total Time	Self Time	Total Elapsed Time	Self Elapsed Time	Average	Call Count
[loop in multiply1 at multiply.c:55]	2 Assumed dependency present	170.635s	170.635s	22.969s	22.969s	1024	4194304
[loop in multiply1 at multiply.c:54]		170.684s	0.049s	22.969s	0.029s	2048	2048
[loop in init_arr at matrix.c:50]	**2 Potential underutilization of FMA ...**	**0.011s**	**0.011s**	**0.011s**	**0.011s**	**256**	**4096**
ThreadFunction		170.684s	0.000	22.969s	0s		8
multiply1		170.684s	0.00	22.969s	0s		
[loop in multiply1 at multiply.c:53]		170.684s	0.00	22.969s	0s	256	



実際のトリップカウントを確認

ループで実行された時間を知るだけでは不十分

ループの反復回数を知る必要があります

ベクトル化しても安全か?

ループ伝搬依存の解析で正当性を検証

4. ループ伝搬依存の解析

Elapsed time: 34.42s | Vectorized | Not Vectorized | Customize View

Filter: All Modules | All Sources | Loops And Functions | All Threads

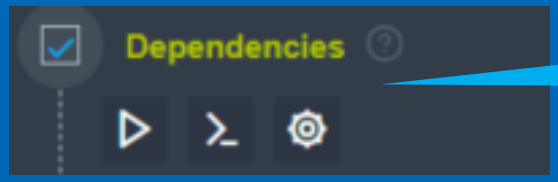
Summary | Survey & Roofline | Refinement Reports

ROOFLINE

Function Call Sites and Loops	Performance Issues	CPU Time Total Time	CPU Time Self Time	Type	Why No Vectorization?
[loop in multiply1 at multiply.c:55]	2 Assumed ...	172.445s	172.445s	Scalar	vector dependence prevents vectorization
[loop in multiply1 at multiply.c:54]		172.490s	0.045s	Scalar	outer loop was not auto-vectorized: consider using
[loop in init_arr at matrix.c:80]	Potential u ...	0.019s	0.019s	Vectorized (Bo...	
ThreadFunction		172.490s	0.000s	Function	
multiply1		172.490s	0.000s	Function	

依存性解析を行うループを選択してボタンをクリック！

ベクトルの依存関係はベクトル化を妨げます



メモリー・アクセス・パターン (MAP) 解析

アクセスパターンを理解

5. メモリー・アクセス・パターンの解析

一意なストライドアクセス

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

一定ストライドアクセス

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

可変ストライドアクセス

```
for (i=0; i<N; i++)
A[B[i]] = C[i]*D[i]
```

Summary | Survey & Roofline | Refinement Reports

Site Location	Loop-Carried Dependencies	Strides Distribution	Access Pattern	Footprint Estimate		
				Max. Per-Instruction Addr. Range	First Instance Site Footprint	Simulated Me
[loop in multiply1 at multiply.c:55]	WAW:1	67% / 33% / 0%	Mixed Strides	31MB	31MB	0B

```
53      for(i=tidx; i<msize; i=i+numt) {
54          for(j=0; j<msize; j++) {
55              for(k=0; k<msize; k++) {
56                  c[i][j] = c[i][j] + a[i][k] * b[k][j];
57              }
```

Memory Access Patterns Report | Dependencies Report | Recommendations

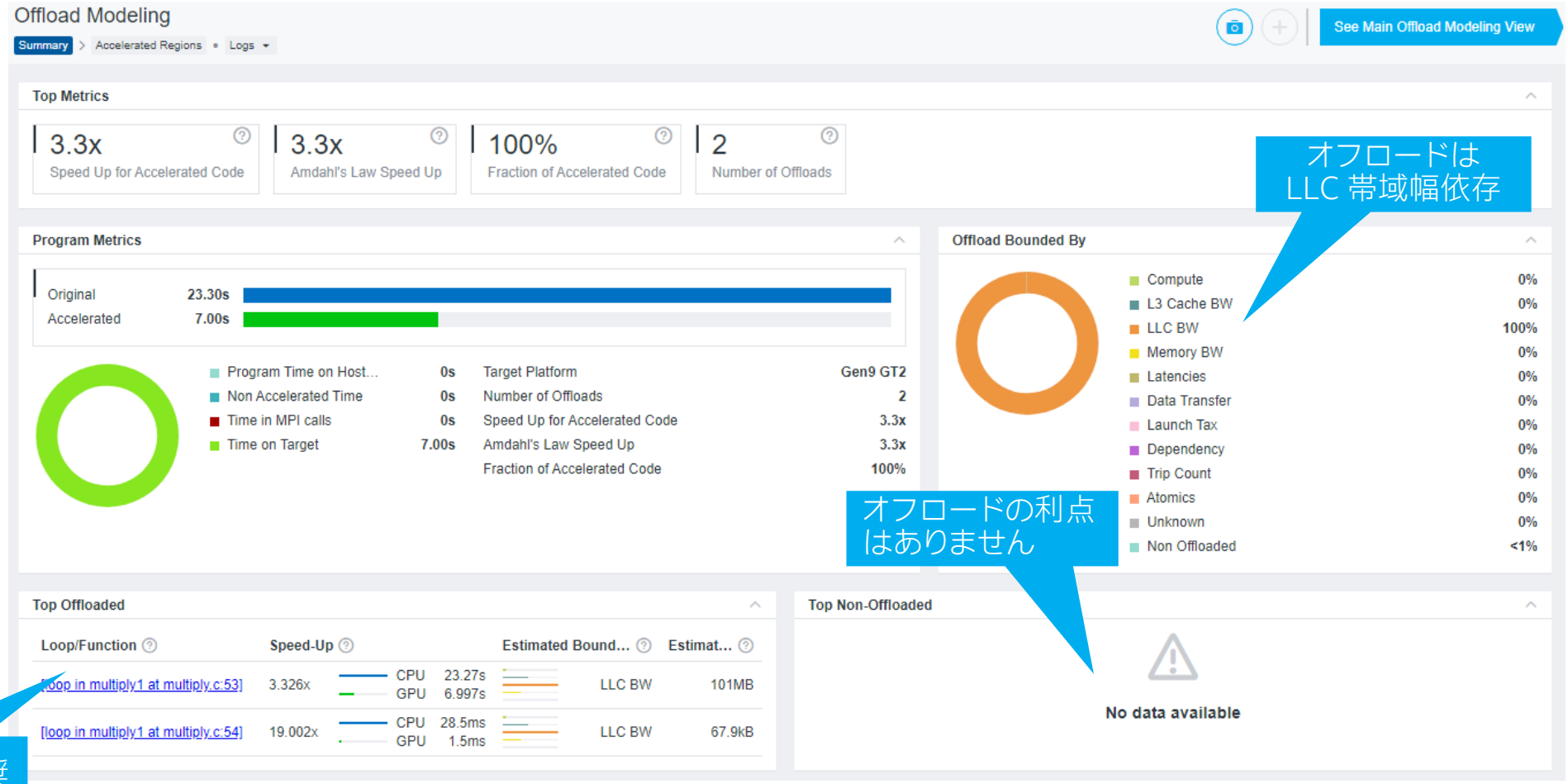
ID		Stride	Type	Source	Nested Function	Variable references	Max. P
P1		2; 4096	Constant stride	multiply.c:56		block 0xcf86020 allocated at matrix.c:147, block 0xef9b020 allocated at matrix.c:152	31MB
P2			Parallel site information	multiply.c:55			
P4		0	Uniform stride	multiply.c:55			4B
P5		0	Uniform stride	multiply.c:56		block 0x10fa8020 allocated at matrix.c:157	8B

```
54          for(j=0; j<msize; j++) {
55              for(k=0; k<msize; k++) {
56                  c[i][j] = c[i][j] + a[i][k] * b[k][j];
57              }
58          }
```

インテル® Advisor による オフロードのモデル化

インテル® Advisor - オフロードのモデル化

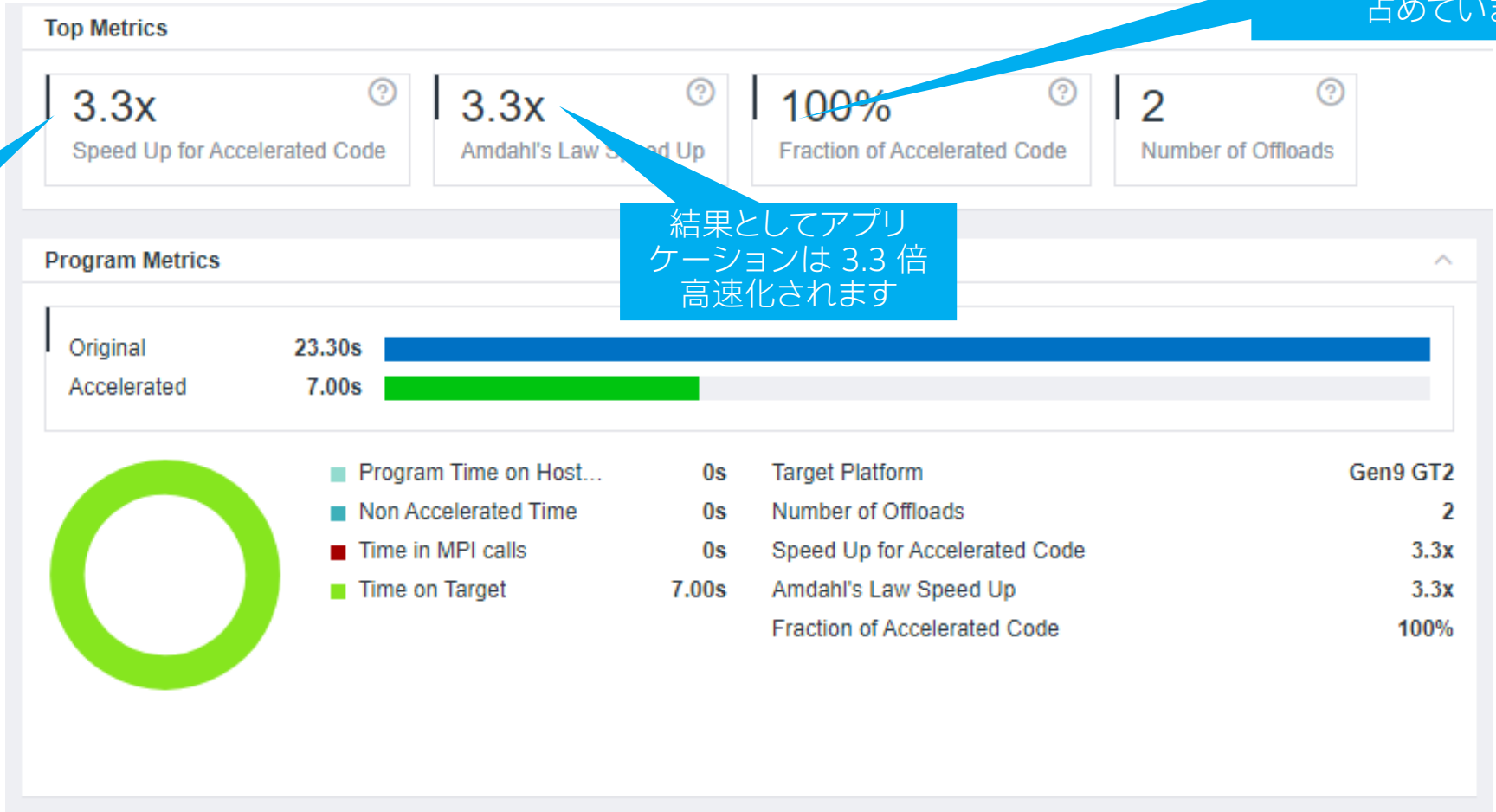
- CPU または GPU で実行 - GPU 向けの予測
- 利用可能なアクセラレーターで実行するコードセクションの定義を支援
- アクセラレーターでの性能を予測します



インテル® Advisor - オフロードのモデル化

オフロードの利点があるコードを特定

ループはアプリケーションの実行時間の 100% を占めています



ループは GPU で CPU の 3.3 倍高速に実行されます

結果としてアプリケーションは 3.3 倍高速化されます

上位オフロード領域の解析

各ループをモデル化して詳しい説明を示します

- タイミング (合計時間、アクセラレーター上の時間、スピードアップ)
- オフロードメトリック (オフロードのデータ転送コスト)
- メモリー・トラフィック (DRAM、L3、L2、L1)、トリップカウント
- アクセラレーターで実行すべきコード領域をハイライト

multiply.c の 53 行目にあるループをオフロードすることを推奨

- LLC 帯域幅依存
- 予測される GPU での実行時間は 6.997 秒
- 101MB のデータを転送

Offload Modeling

Summary > Accelerated Regions • Logs

3.3x
Speed Up for Accelerat

Fraction of Accelerated Code

2
Number of Offloads

CPU+GPU

Loop/Function	Measured		Basic Estimated Metrics		Estimated Bounded By			Estimated Data Transfer With Reuse
	Time	Speed-Up	Time	Offload Summary	Throughput	Taxes With Reuse	Latencies	
▼ [loop in multiply1 at multiply.c:53]	23.27s	3.326x	6.997s	Offloaded	LLC BW 6.997s L3 BW 3.215s	Launch Tax < 0.1ms All Taxes < 0.1ms	Load < 0.1ms	Read 101MB Write 0B
▼ [loop in multiply1 at multiply.c:54]	23.27s							
[loop in multiply1 at multiply.c:55]	23.26s							
▼ [loop in multiply1 at multiply.c:54]	28.5ms	19.002x	1.5ms	Offloaded	LLC BW 1.5ms L3 BW 0.7ms	Launch Tax < 0.1ms All Taxes < 0.1ms	Load < 0.1ms	Read 67.9kB Write 0B
[loop in multiply1 at multiply.c:55]	19.7ms							

上位オフロード領域の解析

ループメトリックはソースおよびコールツリーに関係します

Source | Top-Down | Recommendations

Loop/Function	Measured	Basic Estimated Metrics			Estimated Bounded By			Estimated Data Transfer With Reuse
	Time	Speed-Up	Time	Offload Summary	Throughput	Taxes With Reuse	Latencies	
Total	23.28s							
func@0x4b2e8759	23.27s							
func@0x4b2e8775	23.27s							
BaseThreadInitThunk	23.27s							
ThreadFunction	23.27s							
multiply1	23.27s							
[loop in multiply1 at multiply.c:53]	23.27s	3.326x	6....	Offloaded	LLC... 6.... L3 ... 3.2...	Launch Tax < 0.1ms All Taxes < 0.1ms	L... < 0....	Read 101MB Write 0B
_scrt_common_main_seh	98.5ms							

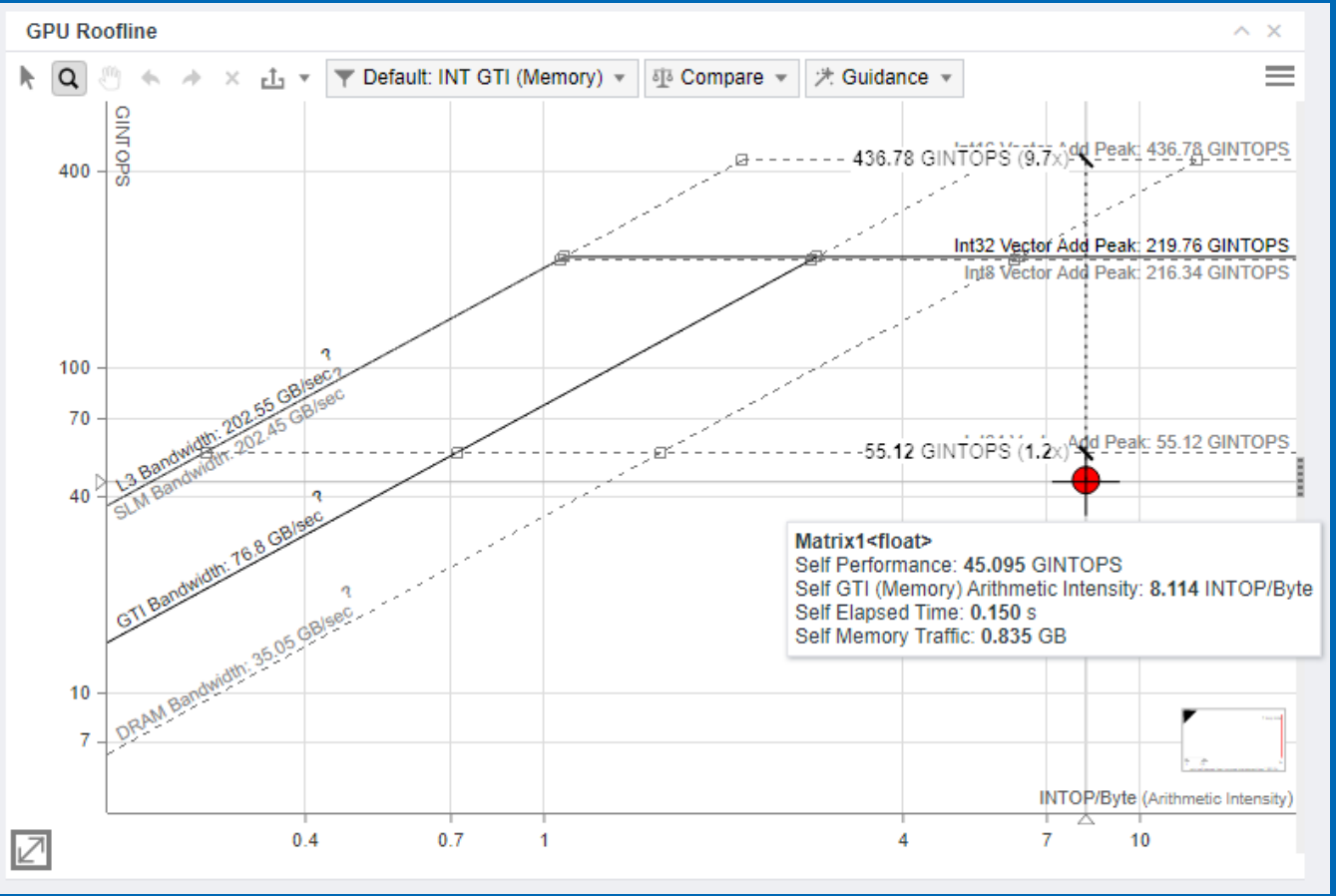
インテル® Advisor による GPU ルーフライン

効果的な最適化の戦略を特定

インテル® Advisor - GPU ルーフライン

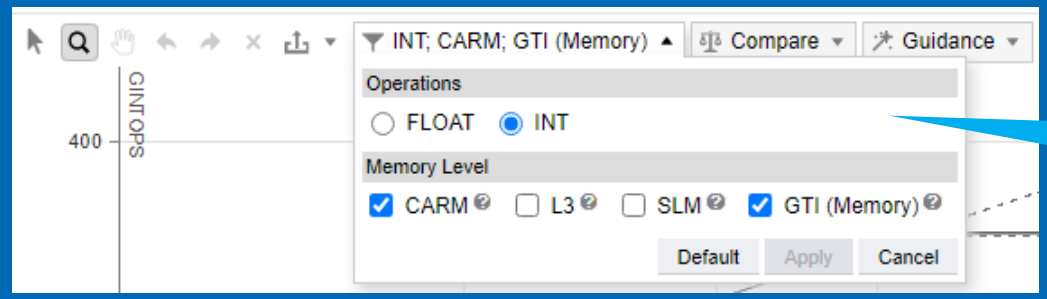
GPU ルーフラインのパフォーマンス詳細

- パフォーマンスの問題があるループをハイライト
- ボトルネックの原因を表示
- 最適化の次のステップを提案
- 各ループのパフォーマンス改善の「余地」を表示
  - -改善可能なループは?
  - -改善の価値があるループは?

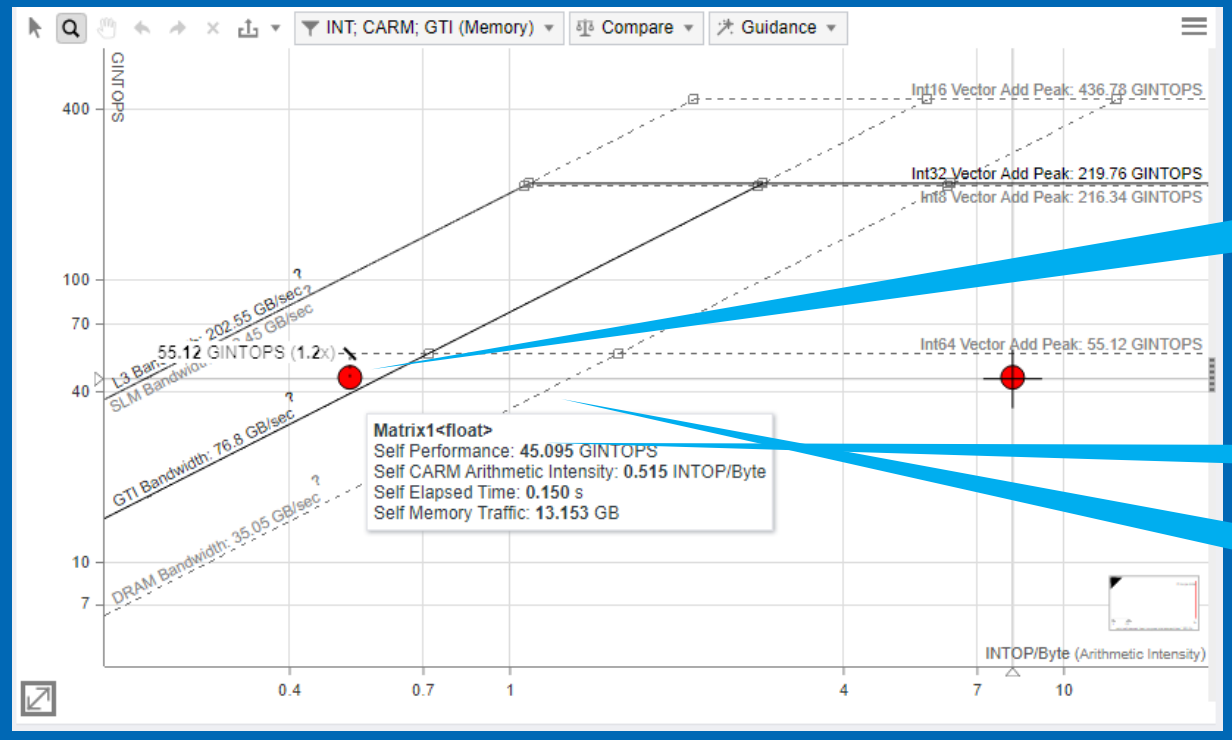


効果的な最適化の戦略を特定

インテル® Advisor - GPU ルーフライン



表示レベルを設定



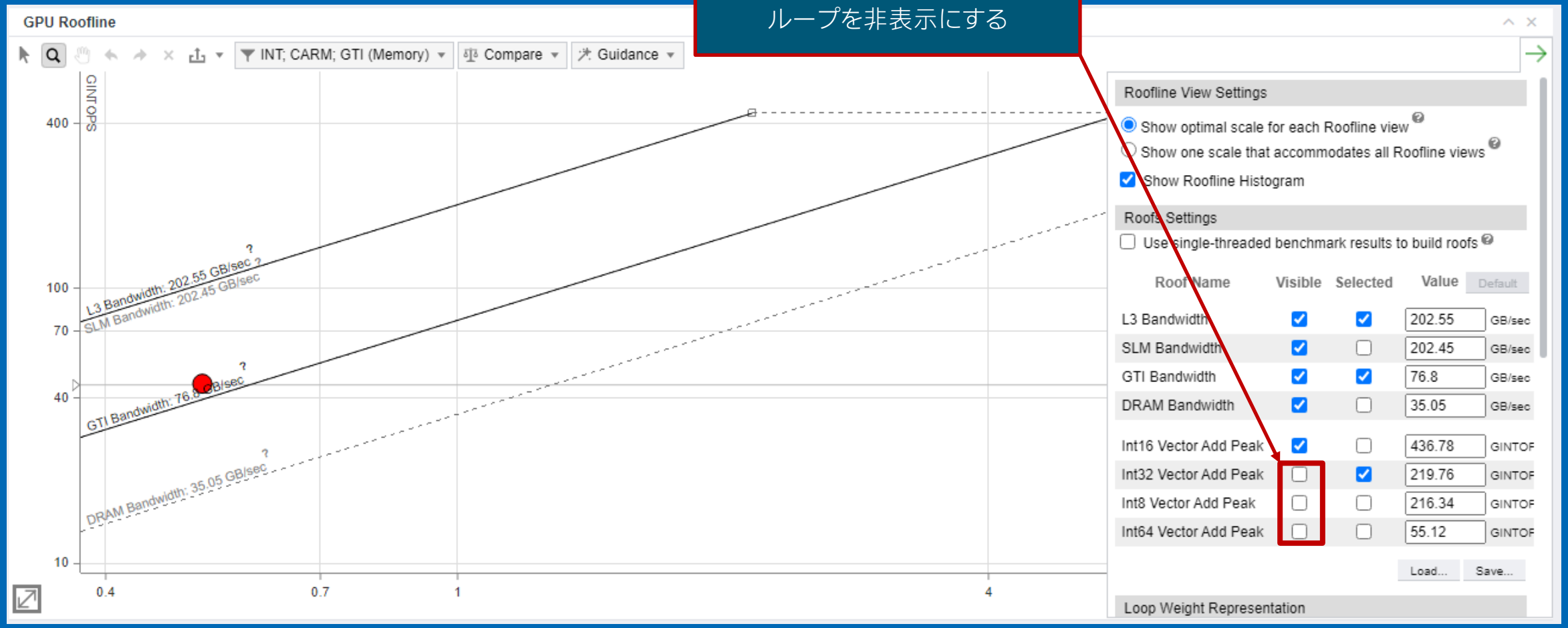
各ループのパフォーマンス改善の余地を表示

ボトルネックの可能性

最適化の次のステップを提案

必要なループだけを表示するようにカスタマイズ

右上隅をクリックして必要のないループを非表示にする



メトリックのサマリー

スナップショットを生成

レポートタブを切り替え

GPU ルーフライン・グラフをカスタマイズ

計算タスクの GPU パフォーマンス

詳細、GPU ソース、GPU アセンブリー情報を表示

GPU Roofline Insights

Summary > GPU Roofline Regions • Logs

53.0% FPU Utilization | 99.4% EU Threading Occupancy | 1.639 EU IPC Rate

U Roofline

INT; CARM; GTI (Memory) | Compare | Guidance

GINTOPS

Int16 Vector Add Peak: 436.78 GINTOPS

L3 Bandwidth: 202.55 GB/sec

SLM Bandwidth: 202.45 GB/sec

GTI Bandwidth: 76.8 GB/sec

DRAM Bandwidth: 35.05 GB/sec

INTOP/Byte (Arithmetic Intensity)

Self Elapsed Time: 0.150 s

Point Info

Matrix1<float>

Self Performance: 45.095 GINTOPS

Self GTI (Memory) Arithmetic Intensity

Bounded by: L3 Bandwidth

Self Elapsed Time: 0.150 s

Self Memory Traffic: 0.835 GB

Copy To Clipboard

Memory Metrics

Impacts: L3 85%, GTI 14%

Shares: L3 13.152GB, GTI 0.835GB

Details | GPU Source | GI

Matrix1<float>

SUMMARY

Elapsed Time 0.15s | GINTOP | GFLOPS

Work Size 1024 x 1024 | Local 256 x 1

ROOFLINE

Bounded by Int32 Vector Add Peak

Int32 Vector Add Peak

(↑2.3x)

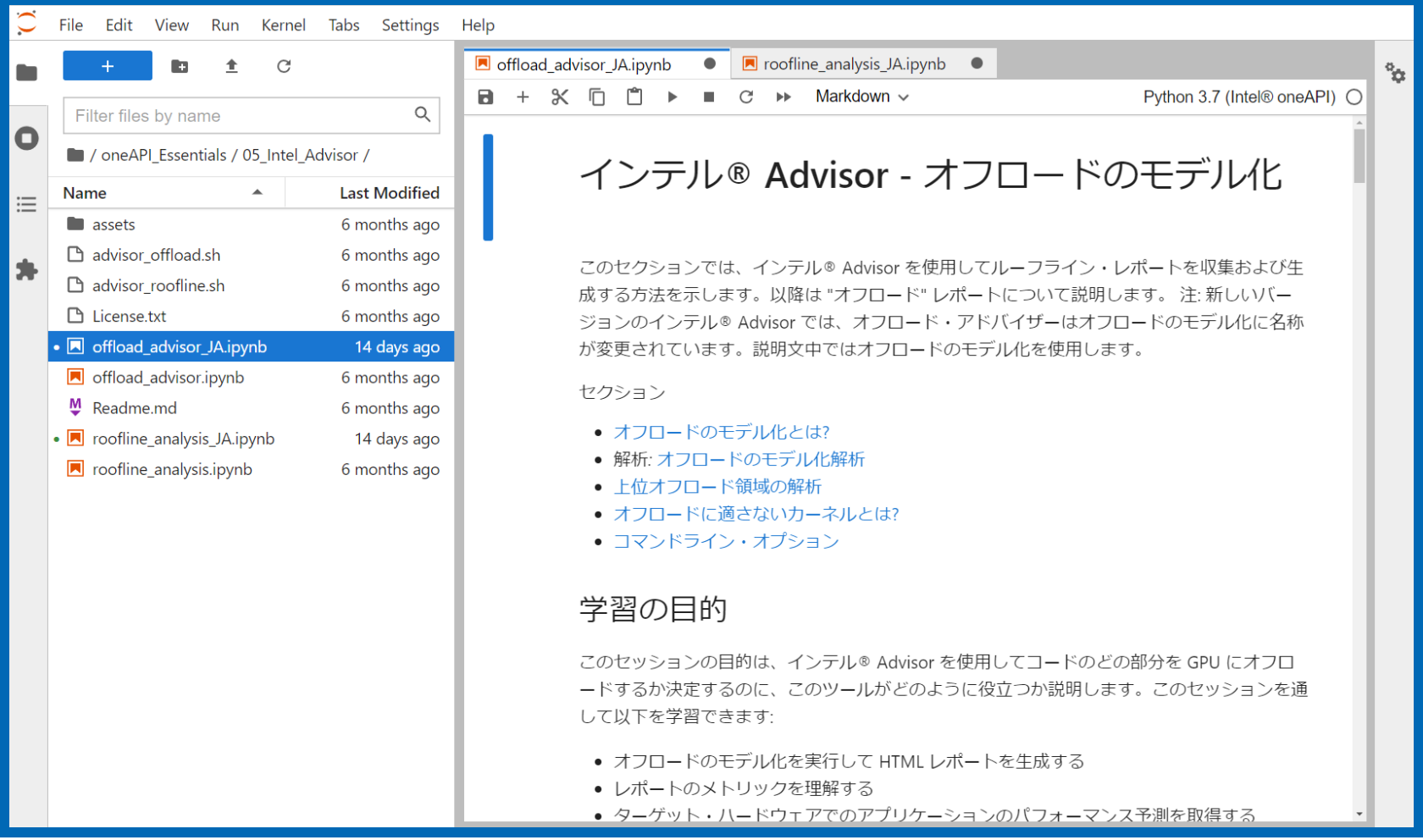
GPU

Compute Task	Elapsed Time	GPU Compute Performance						Work Size		Compute Task Purpose	Compute Task
		GFLOPS	GINTOPS	FP AI	INT AI	GFLOP	GINTOP	Global	Local		
[Outside any task]	3.213s	0.000	0.000	0.000	0.000	0.000	0.000			[Unknown]	0s
zeCommandListAppendMemoryCopyRegion	0.002s	0.000	0.000	0.000	0.000	0.000	0.000			Transfer Out	0.002s
zeCommandListAppendBarrier	0.000s	0.000	0.000	0.000	0.000	0.000	0.000			Synchroniz...	0.000s
Matrix1<float>	0.150s	14.298	45.095	2.573	8.114	2.147	6.773	1024 x 1024	256 x 1	Compute	0.150s

インテル® DevCloud のチュートルアル

インテル® DevCloud のチュートルアル

oneAPI Essentials/05_Intel_Advisor/



法務上の注意書きと最適化に関する注意事項

本資料には、開発中の製品、サービスおよびプロセスについての情報が含まれています。本資料に含まれる情報は予告なく変更されることがあります。 インテル® テクノロジーの機能と利点はシステム構成によって異なり、対応するハードウェアやソフトウェア、またはサービスの有効化が必要となる場合があります。詳細については、各システムメーカーまたは販売店にお問い合わせいただくか、http://www.intel.co.jp/ を参照してください。

上記のベンチマーク結果は、追加のテストによって変更が必要になる可能性があります。結果は、テストに使用される特定のプラットフォーム構成や作業負荷に依存します。個々のユーザーのコンポーネント、コンピューター・システム、作業負荷では同様の結果が得られない可能性があります。結果は、必ずしもほかのベンチマークを代表するものではなく、ほかのベンチマークでは、結果が異なることがあります。

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。詳細については、www.intel.com/benchmarks (英語) を参照してください。

本資料に掲載されている情報は現状のまま提供され、いかなる保証もいたしません。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を許諾するための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。

© 2021 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴ、その他のインテルの名称やロゴは、Intel Corporation またはその子会社の商標です。
* 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。

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。

改訂 #20110804

intel®