

<https://techdecoded.intel.io/essentials/profile-dpc-and-gpu-workload-performance/>



# DPC++ と GPU ワークロードの パフォーマンスをプロファイル

インテル® VTune™ プロファイルラー

インテルコーポレーション  
テクニカル・コンサルティング・エンジニア  
Vladimir Tsymbal



# 内容

- GPU プログラミング・モデルの概要
- インテル® VTune™ プロファイラーの GPU 解析の概要
- オフロード・パフォーマンスのチューニング
- GPU 計算/メディア・ホットスポット
- まとめ

# インテルの GPU とプログラミング・モデル

## ■ Gen9

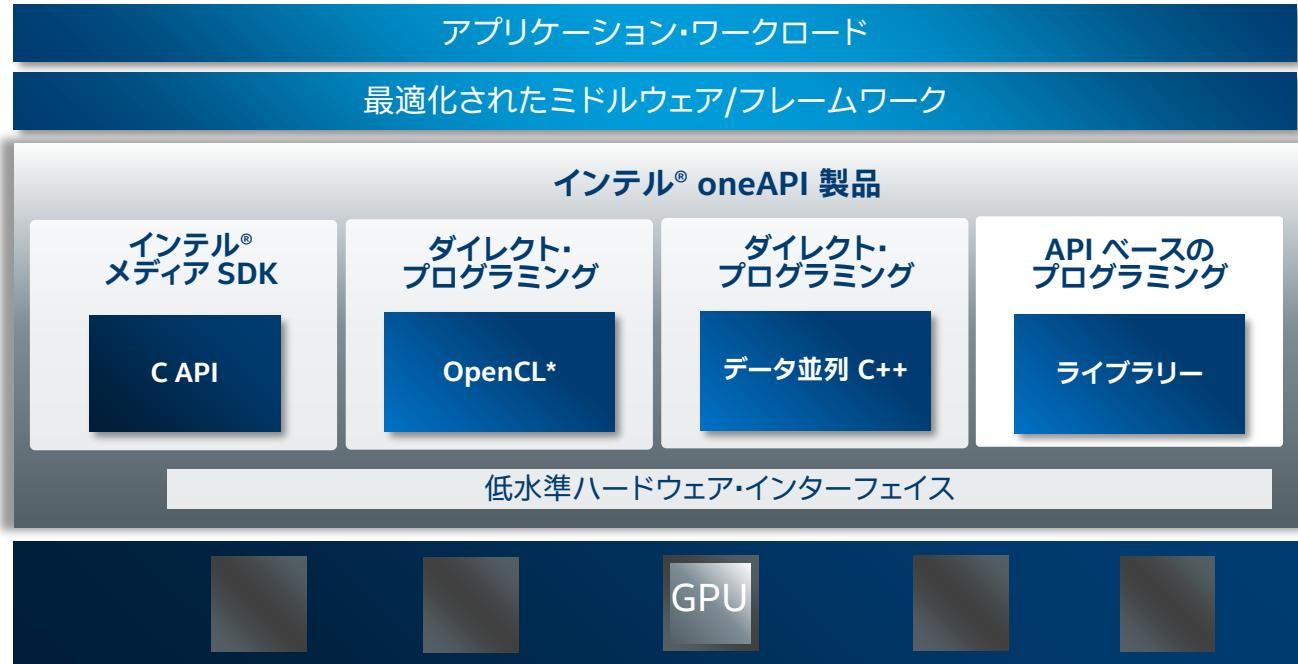
- モバイル、デスクトップ、ワークステーションで一般的

## ■ Gen11

- Ice Lake<sup>†</sup> CPU 搭載のモバイル・プラットフォーム

## ■ Gen12

- Tiger Lake<sup>†</sup> CPU 搭載のモバイル、Rocket Lake<sup>†</sup> CPU 搭載のデスクトップ



詳細は、[software.intel.com/oneapi](https://software.intel.com/oneapi) (英語) を参照

<sup>†</sup> 開発コード名

# インテル® VTune™ プロファイラーの GPU 解析の概要

# GPU アプリケーション解析

## GPU 計算/メディア・ホットスポット

- ホストと GPU を可視化
- HW イベントベースのパフォーマンス・チューニング手法
- タイムライン表示と集計表示を提供

## GPU インカーネル・プロファイル

- ソース/命令レベルの GPU プロファイル
- SW インストルメンテーション
- 2 つのモード: 基本ブロック・レイテンシーとメモリー・アクセス・レイテンシー

GPU 占有率とプロファイルするカーネルを特定して、細粒度でカーネルをチューニング

# GPU 解析: ホットスポット

- [Summary (サマリー)] ページの各セクションの説明に従うだけ
- HW メトリックに基づくチューニング手法

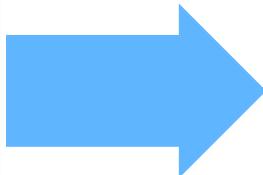
## GPU Usage <sup>②</sup>: 0.6% ↗

Use this section to understand whether the GPU was utilized properly and which of the engines were utilized. Identify the amount of gaps in the GPU utilization that potentially could be loaded with some work. This metric is calculated for the engines that had at least one piece of work scheduled to them.

### GPU Usage

GPU Usage breakdown by GPU engines and work types.

GPU Engine / Packet Type	GPU Time (ms)	(%) <sup>③</sup>
Render and GPGPU	1.146s	0.6% ↗
Unknown	0.888s	0.5%
GHAL3D	0.249s	0.1%
OpenCL	0.009s	0.0%



## EU Array Stalled/Idle <sup>②</sup>: 94.4% ↗ of Elapsed time

Analyze the average value of EU Array Stalled/Idle metric and identify why EUs were waiting for resources instead of doing computations. This metric is critical for compute-bound applications. Explore typical reasons for this kind of inefficiency listed below.

### GPU L3 Bandwidth Bound <sup>②</sup>: 0.5% of peak value

### DRAM Bandwidth Bound <sup>②</sup>: 0.0% of Elapsed time

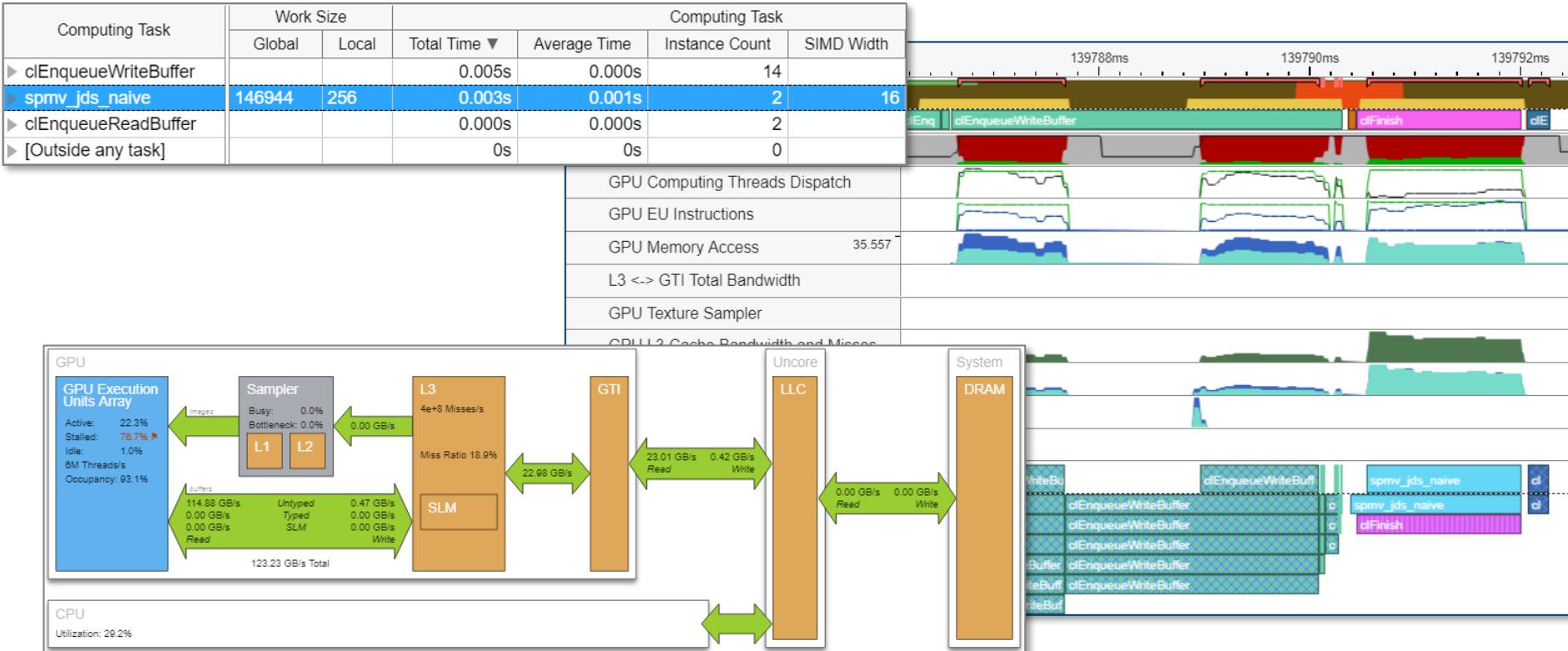
### Occupancy <sup>②</sup>: 25.8% ↗ of peak value

Identify too large or too small computing tasks with low occupancy that make the EU array idle while waiting for the scheduler. Note that frequent SLM accesses and barriers may affect the maximum possible occupancy.

### Hottest GPU Computing Tasks with Low Occupancy

### Sampler Busy <sup>②</sup>: 40.6% of peak value

# GPU 解析: 集計表示とタイムライン表示



# GPU 解析: インカーネル・プロファイル

## GPU カーネル実行の解析

- メモリー・レイテンシーや非効率的なカーネル・アルゴリズムを検出
- OpenCL\* ソースコードとアセンブリーコードのホットスポットを確認
- DMA パケット実行を解析
  - パケットキューの深さヒストグラム
  - パケットの持続時間ヒストグラム
- GPU 側のコールスタック

Source	Estimated GPU Cycles
256 #ifdef USE_IMAGE_STORAGE	
257 // Read the node information from the image.	0.2%
258 const ushort inx = (nodeData >> 16) * 7;	
259 const ushort iny = (nodeData & 0xffff);	
260 const float4 bboxes_minX = as_float4(read_imageui(inx, iny, 0));	0.8%
261 const float4 bboxes_maxX = as_float4(read_imageui(inx, iny, 1));	0.7%
262 const float4 bboxes_minY = as_float4(read_imageui(inx, iny, 2));	0.7%
263 const float4 bboxes_maxY = as_float4(read_imageui(inx, iny, 3));	0.7%
264 const float4 bboxes_minZ = as_float4(read_imageui(inx, iny, 4));	0.7%
265 const float4 bboxes_maxZ = as_float4(read_imageui(inx, iny, 5));	0.7%
266 const int4 children = as_int4(read_imageui(inx, iny, 6));	0.7%
267	
268 const int4 visit = QBVHNode_BBoxIntersect(	13.1%
269     bboxes_minX, bboxes_maxX,	
270     bboxes_minY, bboxes_maxY,	
271     bboxes_minZ, bboxes_maxZ,	

# オフロード・パフォーマンスの チューニング

# GPU オフロード解析

## ヘテロジニアス・アプリケーション

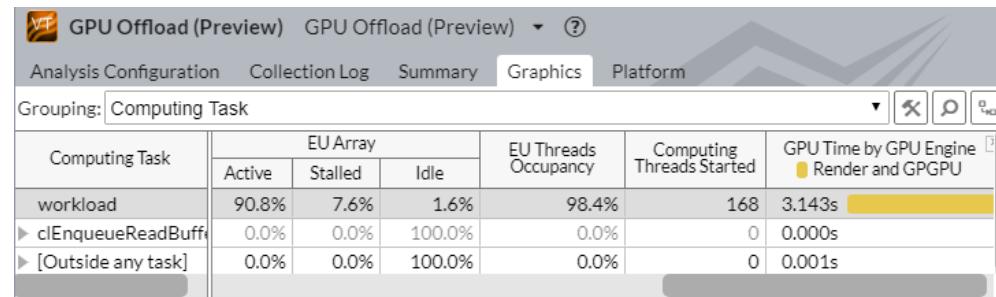
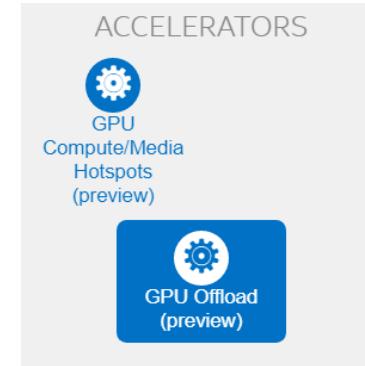
- オフロードモデル: SYCL\*、DPC++、OpenMP\*

## すべての実行リソースに注目

- 各種 CPU および GPU コア上のコード実行を調査
- CPU と GPU アクティビティーを関連付け
- アプリケーションが GPU 依存か、CPU 依存かを特定

## 詳しく解析するカーネルを特定

- タスクレベルの解析
- カーネルの効率
- データ転送レート



# SYCL\* について



SYCL\* はホスト (CPU) のコードにデバイス (GPU など) のコードをインラインで埋め込む共有ソースデザインを実装

- OpenCL\* を使用するアルゴリズムの再利用可能なテンプレート
- ホストとデバイスのコード間のタイプセーフな相互作用

言語拡張 (OpenCL\*) を使用しないシングルソース C++

- OpenCL\* と C++14 標準ベース

標準 C++ コンパイラーでコンパイルして、OpenCL\* デバイスが利用できない場合は CPU にフォールバック

**DPC++ = ISO C++ + Khronos SYCL\* + コミュニティー拡張**

# DPC++ サンプル・アプリケーション

DPC++ は次のような追加機能を提供する SYCL\* 拡張

- 統合共有メモリー (USM)
- ND-range サブグループ
- 順序付きキュー、など

GitHub\* の `oneapi-src/oneAPI-samples` リポジトリーからさまざまなサンプル・アプリケーションを利用可能 (<https://github.com/oneapi-src/oneAPI-samples> (英語))

- 例: `matrix_multiply_vtune`
- 複数のカーネルを選択して行列乗算を実行
- 完全なオフロードの例ではない

<https://github.com/oneapi-src/oneAPI-samples>

```
void multiply1(int msize, int tidx, int numt, TYPE a[][NUM], TYPE b[][NUM], TYPE c[][NUM], TYPE t[][NUM]) {  
    int i, j, k;  
  
    default_selector device;  
    queue q(device, exception_handler);  
  
    range<2> matrix_range{NUM, NUM};  
  
    buffer<TYPE, 2> bufferA((TYPE*)a, matrix_range);  
    buffer<TYPE, 2> bufferB((TYPE*)b, matrix_range);  
    buffer<TYPE, 2> bufferC((TYPE*)c, matrix_range);  
  
    q.submit([&](cl::sycl::handler& h) {  
        auto accessorA = bufferA.get_access<sycl_read>(h);  
        auto accessorB = bufferB.get_access<sycl_read>(h);  
        auto accessorC = bufferC.get_access<sycl_read_write>(h);  
  
        h.parallel_for<class Matrix1<TYPE>>(matrix_range, [=](cl::sycl::id<2> ind) {  
            int k;  
            for (k = 0; k < NUM; k++) {  
                accessorC[ind[0]][ind[1]] += accessorA[ind[0]][k] * accessorB[k][ind[1]];  
            }  
        });  
    }).wait_and_throw();  
}
```

deviceQueue を宣言

2 次元の範囲を宣言

3 つのバッファーを宣言して初期化

ジョブをキューに送信

バッファーへの 3 つのアクセサーを宣言 2RD, 1 WR.

matrix\_range に対して行列乗算を並列に実行

ind はこの範囲のインデックス

計算を実行: ind[0] は行、ind[1] は列

# GPU 計算/メディア・ホットスポット

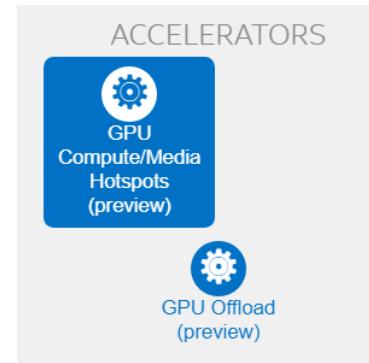
# GPU 計算/メディア・ホットスポット

## GPU 依存解析

- SoC 関連の一部のメトリックも測定される

## GPU のパフォーマンスを最大限に引き出す方法

- 理想的には、最適化された IP ブロックを使用する - パフォーマンス・ライブラリー
- DPC++ のような高レベルのモデルでは GPU アーキテクチャーに合わせてワークロードの調整が可能
- インテル® VTune™ プロファイラーは HW レベルのメトリックを提供するため GPU ブロックの知識が必要



# GPU アーキテクチャー

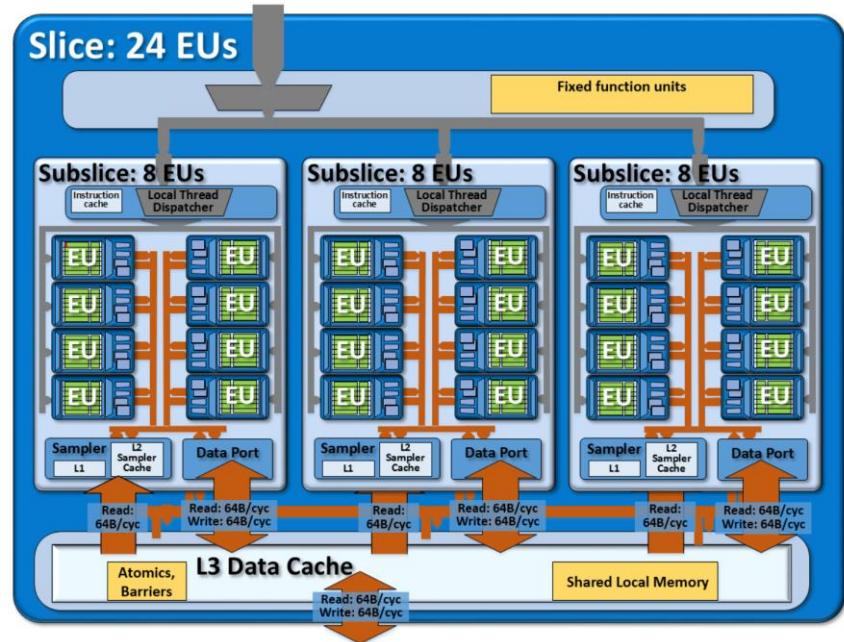
## Collection and Platform Info

### GPU

Name: Intel(R) UHD Graphics 620  
Vendor: Intel Corporation  
Driver: 27.20.100.8187  
EU Count: 24  
Max EU Thread Count: 7  
Max Core Frequency: 1.1 GHz

### GPU OpenCL Info

Version: OpenCL C 2.0  
Max Compute Units: 24  
Max Work Group Size: 256  
Local Memory: 64 KB  
SVM Capabilities: Fine-grained buffer with atomics



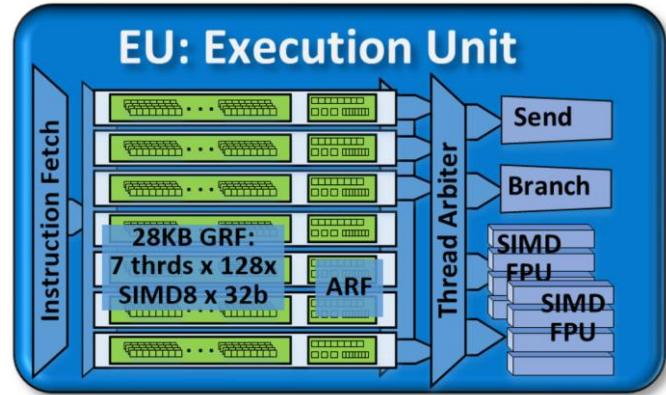
インテル® VTune™ プロファイラーの [Summary (サマリー)] ページから  
GPU アーキテクチャーの詳細を素早く確認できる

# Gen9 GPU の EU の詳細

## EU の計算アーキテクチャー:

- 24 EU × 7 スレッド = 168 スレッドが利用可能
- 128 GRF (32 バイト、8 つの 32 ビット・データ要素のベクトルとしてアクセス可能)、柔軟なアクセスが可能
- 2 SIMD-4 FPU (32 ビットの FP または INT データ)
- 16 MAD/サイクル—(ADD + MUL) × 2 FPU × SIMD-4
- 2 つの追加ユニット: 分岐と送信

EU を最大限に活用することが目標



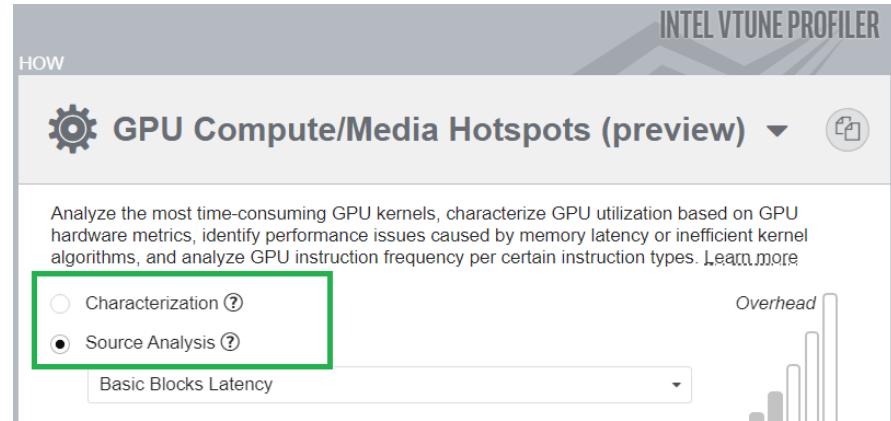
# インテル® VTune™ プロファイラーの GPU 計算/ メディア・ホットスポット解析

次のいずれかの GPU 解析設定を選択

- **[Characterization (特性化)]** – GPU エンジンの使用状況、効率、ストールを監視
- **[Source Analysis (ソース解析)]** – GPU カーネル内のパフォーマンスに重大な影響を与えるブロックとメモリーアクセスの問題を特定

最適化の方針:

- EU を最大限に有効活用
- SIMD を最大限に活用
- メモリーの問題による EU ストールを最小限に抑える



# EU 効率とメモリー問題の解析

## [Characterization (特性化)] オプションを使用

- EU アクティビティー: EU アレイアクティブ、EU アレイストール、EU アレイアイドル、開始された計算スレッド、コア周波数

## [Overview (概要)] または [Compute Basic (基本計算)] メトリックを選択

- 追加のメトリック: メモリー読み取り/書き込み帯域幅、GPU L3 ミス、型付きメモリー読み取り/書き込みトランザクション

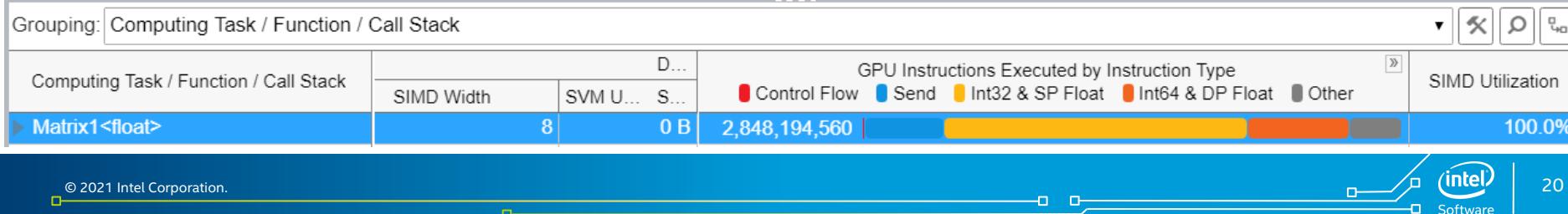


Grouping: Computing Task									
Computing Task	EU Array			EU Threads Occupancy	Computing Threads Started	EU Instructions			L3 Bandwidth, GB/sec
	Active	Stalled	Idle			IPC Rate	2 FPUs active	Send active	
Matrix1<float>	62.6%	37.4%	0.0%	95.1%	131,040	1.583	34.9%	6.8%	89.763

# GPU 命令実行の解析

## [Dynamic Instruction Count (動的命令数)] プリセットを使用

- カーネルで実行された命令の内訳
- 命令グループ
  - 制御フロー
  - 送信
  - 同期
  - Int16 & HP Float | Int32 & SP Float | Int64 & DP Float
  - その他



# ソースコードの解析

## [Source Analysis (ソース解析)] オプションを使用

- 注目するカーネルの基本ブロック・レイテンシーやメモリー・レイテンシーの問題を解析
- [Source (ソース)] と [Assembly (アセンブリー)] ペインを並べて表示

Source		Assembly		GPU Instructions Exec		Assembly grouping: Address	
1	void __kernel sin_cos(__global float*			Control Flow	GPU Instructions Exec	A... ▲ S...	Assembly
2	{			Send	Int32 & SF	0x598 7	cmp (16 M0) (gt) f0.0 null<1>; 16,384 50.0%
3	size_t i = get_global_id(0);					0x5a8 7	cmp (16 M16) (gt) f0.0 null<1>; 16,384 50.0%
4						0x5b8 7	(~f0.0) if (32 M0) bb_12 bb_2; 16,384
5	if (i % 2)					0x5c8	Block 11:
6	{					0x5c8 7	mad (16 M0) r2.0<1>; 313 50.0%
7	data[i] *= sin((float)i);	7,152,845				0x5d0 7	mad (16 M16) r4.0<1>; 313 49.8%
8	}					0x5d8 7	shl (16 M0) r12.0<1>; 313 50.0%
9	else					0x5e8 7	shl (16 M16) r18.0<1>; 313 49.8%
10	{					0x5f8 7	add (16 M0) r2.0<1>; 313 50.0%
11	data[i] *= cos((float)i);	7,057,665				0x608 7	add (16 M16) r4.0<1>; 313 49.8%
12	}					0x618 7	mad (16 M0) r120.0<1>; 313 50.0%
						0x620 7	mad (16 M16) r6.0<1>; 313 49.8%
						0x628 7	mad (16 M0) r118.0<1>; 313 50.0%
						0x638 7	mad (16 M16) r118.0<1>; 313 49.8%

# まとめ

- インテル® VTune™ プロファイラーは強力な GPU パフォーマンス解析ツール
- 新しいオフロードモデル SYCL\*/DPC++/OpenMP\* をサポート
- GPU パフォーマンスのチューニングを支援する 2 つのプレビュー解析タイプ: GPU オフロードと GPU 計算/メディア・ホットスポット
- タスク・スケジュールからアセンブリー命令レインテンシーまで、さまざまなパフォーマンス・メトリックにより、高レベルの推定値と低レベルの測定値を提供

# 関連情報

インテル® VTune™ プロファイラー

<https://www.xlsoft.com/jp/products/intel/vtune/index.html>

GPU オフロード解析:

<https://www.isus.jp/wp-content/uploads/vtune/2020/help/index.htm>

GPU 計算/メディア・スポット解析:

<https://www.isus.jp/wp-content/uploads/vtune/2020/help/index.htm>

DPC++ 仕様:

<https://spec.oneapi.com/versions/latest/elements/dpcpp/source/index.html> (英語)

GitHub\* で公開されているサンプル・アプリケーション

<https://github.com/oneapi-src/oneAPI-samples> (英語)

# 法務上の注意書き

最適化に関する注意事項: インテル® コンパイラでは、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。

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。さらに詳しい情報を知りになりたい場合は、<https://www.intel.com/benchmarks/> (英語) を参照してください。

ここに記載されているすべての情報は、予告なく変更されることがあります。

インテルのテクノロジーを使用するには、対応したハードウェア、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。

結果は推定またはシミュレーションしたものです。

絶対的なセキュリティーを提供できる製品またはコンポーネントはありません。

インテルは、サードパーティのデータについて管理や監査を行っていません。ほかの情報も参考にして、正確かどうかを評価してください。

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。

インテルは、明示されているか否かにかかわらず、いかなる保証もいたしません。ここにいう保証には、商品適格性、特定目的への適合性、および非侵害性の默示の保証、ならびに履行の過程、取引の過程、または取引での使用から生じるあらゆる保証を含みますが、これらに限定されるわけではありません。

© 2021 Intel Corporation. 無断での引用、転載を禁じます。

Intel、インテル、Intel ロゴ、VTune は、アメリカ合衆国および / またはその他の国における 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。

