インテル® VTune™ プロファイラーの GPU 向け機能の概要

intel®

内容

1 インテル® VTune™ プロファイラーと GPU

2 インテル® VTune™ プロファイラーによる GPU 最適化の概要

3 インテル® VTune™ プロファイラーによる解析例

インテル® oneAPI ツールキット

スケーラブルで高速なアプリケーションを実現

ツールキットの概要

C++、Fortran、OpenMP* および MPI でエンタープライズ、クラウド、HPC、AI 向けのハイパフォーマンスでスケーラブルな並列アプリケーションを開発するためにインテル® oneAPI ベース・ツールキットに追加するツールキット

対象ユーザー

- OEM/ISV
- C++、Fortran、OpenMP*、MPI 開発者

ツールキットの重要性

- インテル® Xeon® プロセッサー、インテル® Core™ プロセッサーおよびインテル® アクセラレーターのパフォーマンスを向上
- 少ない労力で、高速かつ安定した、スケーラブルな業界標準の並列コードを作成

詳細: intel.com/oneAPI-HPCKit (英語)

インテル® oneAPI ベース & HPC ツールキット

ダイレクト・プログラミング	API ベースのプログラミング	解析/デバッグツール
インテル® C++ コンパイラー・クラシック	インテル® MPI ライブラリー	インテル® Inspector
インテル® Fortran コンパイラー・クラシック	インテル® oneAPI DPC++ ライブラリー (インテル® oneDPL)	インテル® Trace Analyzer & Collector
ベータ版インテル® Fortran コンパイラー	インテル® oneAPI マス・カーネル・ライブラリー (インテル® oneMKL)	インテル® Cluster Checker
インテル® oneAPI DPC++/C++ コンパイラー	インテル® oneAPI データ・アナリティクス・ライブラリー (インテル® oneDAL)	インテル® VTune™ プロファイラー
インテル® DPC++ 互換性ツール	インテル® oneAPI スレッディング・ビルディング・ブロック (インテル® oneTBB)	インテル® Advisor
インテル® ディストリビューションの Python*	インテル® oneAPI ビデオ・プロセシング・ライブラリー (インテル® oneVPL)	インテル® ディストリビューションの GDB
oneAPI ベース・ツールキット用 インテル® FPGA アドオン	インテル® oneAPI コレクティブ・コミュニケーション・ライブラリー (インテル® oneCCL)	
	インテル® oneAPI ディープ・ニューラル・ネットワーク・ライブラリー (インテル® oneDNN)	
	インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP)	

■ インテル® oneAPI HPC ツールキット +
■ インテル® oneAPI ベース・ツールキット

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

インテル® VTune™ プロファイラーと GPU

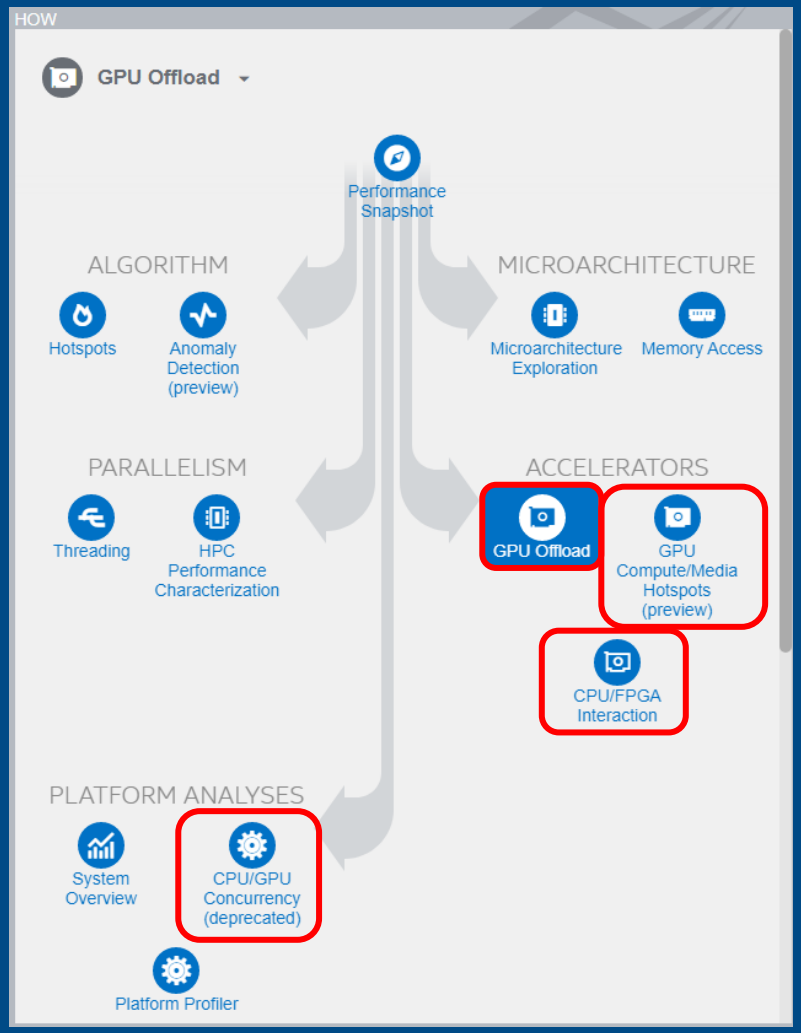
インテル® VTune™ プロファイラー GUI: 概要

- GUI には解析を設定する 3 つのペインがあります
  - **[WHERE (どこを)]** では解析システムを指定します
  - **[WHAT (何を)]** では解析のターゲットを指定します
  - **[HOW (どのように)]** では解析タイプを選択します

インテル® VTune™ プロファイラーのドキュメント:

WHERE (どこを): 解析システム、WHAT (何を): 解析ターゲット および HOW (どのように): 解析タイプ

GPU 向けの解析タイプ



- **GPU Offload**

  レンダリング、ビデオ処理、および計算にグラフィックス処理ユニット (GPU) を使用するアプリケーションをターゲットとします。アプリケーションが CPU または GPU のどちらに依存しているか判断するのに役立ちます

- **GPU Compute/Media Hotspots**

  GPU 依存のアプリケーションを対象とし、コード行ごとの GPU カーネル実行を解析し、メモリー・レイテンシーや非効率なカーネルのアルゴリズムによるパフォーマンスの問題を特定します

- **CPU/FPGA Interaction**

  各 FPGA アクセラレーターの FPGA 利用率を調査し、最も時間を消費する FPGA タスクを特定します

- **CPU/GPU Concurrency (廃止予定)**

インテル® VTune™ プロファイラー CLI: 概要

- CLI には複数レベルのヘルプがあります
  - **vtune -help**
  - **vtune -help collect**
  - **vtune -help collect gpu-offload**
- 収集を実行します
  - **vtune -collect <解析タイプ> <ターゲット>**
- レポートを生成します
  - **vtune -report <レポート名> -r <結果フォルダー>**

Terminal

File Edit View Search Terminal Help

```
[~]$ vtune --help
Intel(R) VTune(TM) Profiler Command Line Tool
Copyright (C) 2009-2020 Intel Corporation. All rights reserved.

 Usage: vtune <-action> [-action-option] [-global-option] [[--]
 target [target options]]

 Type 'vtune -help <action>' for help on a specific action.


Available Actions:

    collect        Choose an analysis type.
    collect-with   Choose a collector.
    command        Issue a command to a running collection.
    finalize       Re-finalize the result.
    help           Display help text.
    import         Create a result directory by importing ...
    report         Generate a report with the specified name.
    version        Display product version.
```

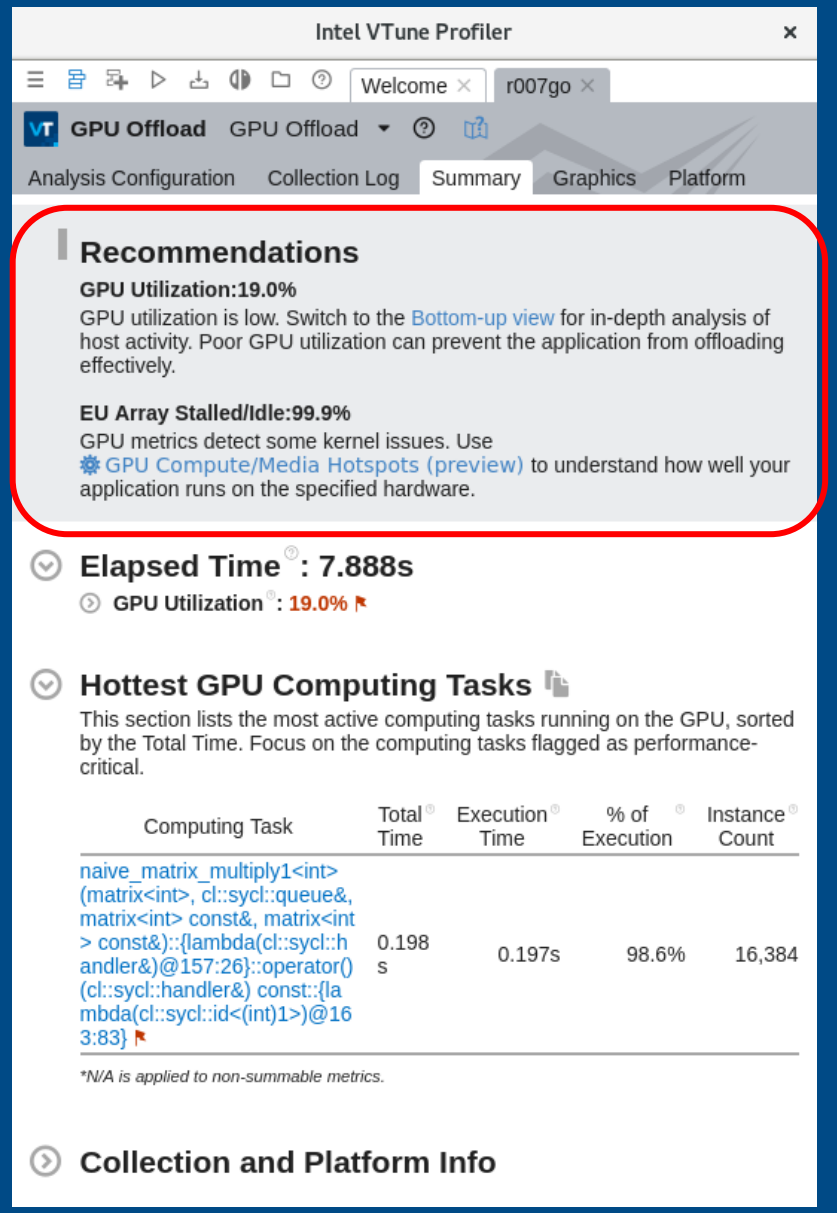
インテル® VTune™ プロファイラーのドキュメント:
コマンドライン・インターフェイス

GPU オフロード

- 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するのに役立ちます
- インテル® メディア SDK ベースのアプリケーションや、OpenCL* および レベル 0 インターフェイスを使用して、インテル® GPU にワークをオフロードする DPC++ および OpenMP* アプリケーションのプロファイルに使用できます

インテル® VTune™ プロファイラーのドキュメント:

GPU オフロード解析



インテル® VTune™ プロファイラーを使用して GPU 使用状況を最適化
GPU オフロード

GPU Offload (Preview) GPU Offload (Preview)

INTEL VTUNE PROFILER

Analysis Configuration | Collection Log | Summary | Graphics | Platform

Elapsed Time: 2.017s

GPU Usage: 47.8%

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	GPU Utilization
Render and GPGPU	0.964s	47.8%
Unknown	0.964s	47.8%

*N/A is applied to non-summable metrics.

Packet Queue Depth Histogram

Packet Duration Histogram

Hottest GPU Computing Tasks

This section lists the most active computing tasks running on the GPU, sorted by the Total Time. Focus on the computing tasks flagged as performance-critical.

Computing Task	Total Time	Total Compute Time	Total Transfer Time (f)
Matrix<float>	3.980s	0.961s	3.019s
clEnqueueReadBufferRect	0.000s	0.000s	0.000s

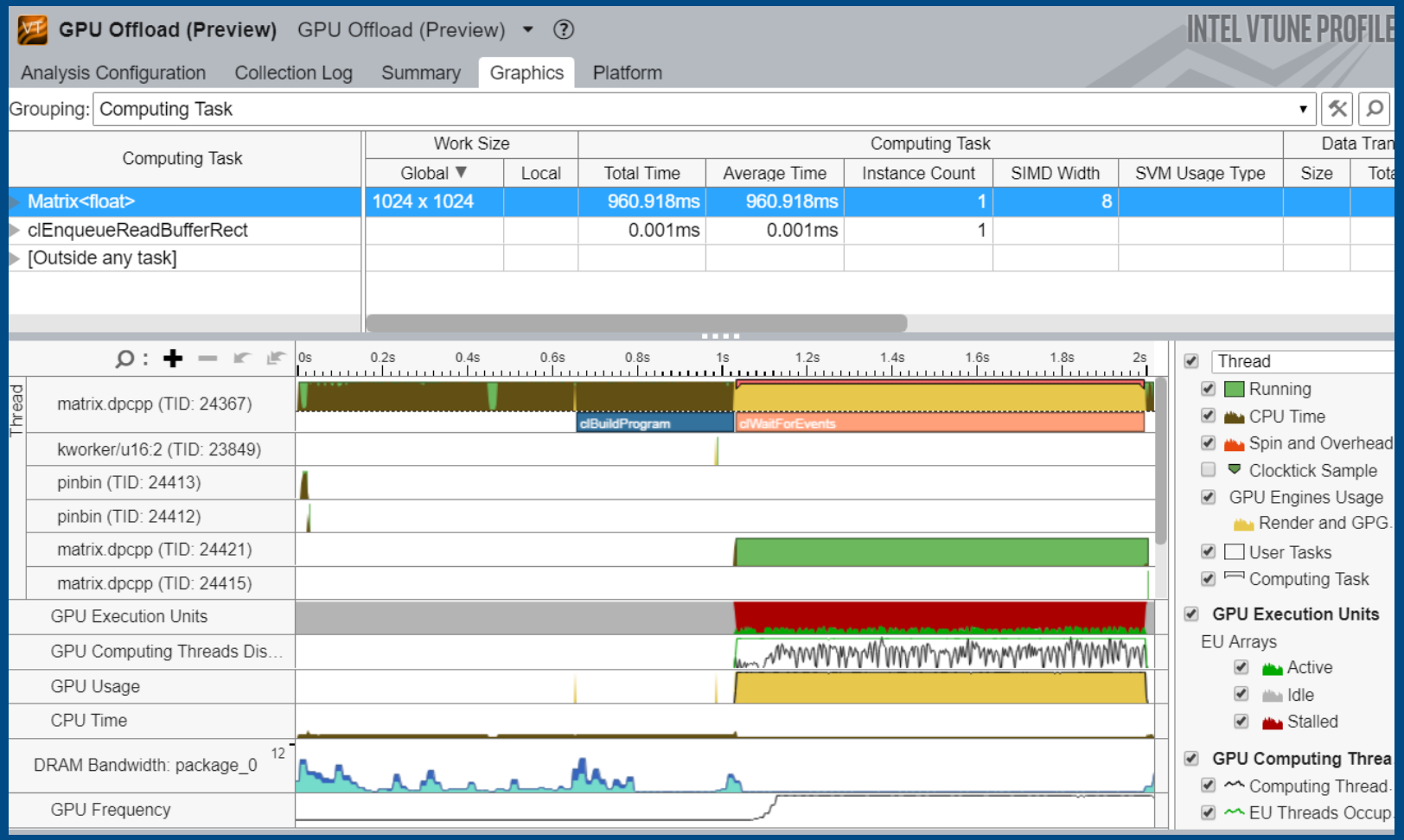
*N/A is applied to non-summable metrics.

この最適化は以下を可能にします:

アプリケーションが DPC++ または OpenCL* カーネルをどのくらい効率良く使用しているか確認します

GPU の使用状況を調査して、すべての時点での GPU エンジンのソフトウェア・キューを解析します

インテル® VTune™ プロファイラーを使用して GPU 使用状況を最適化
GPU オフロード



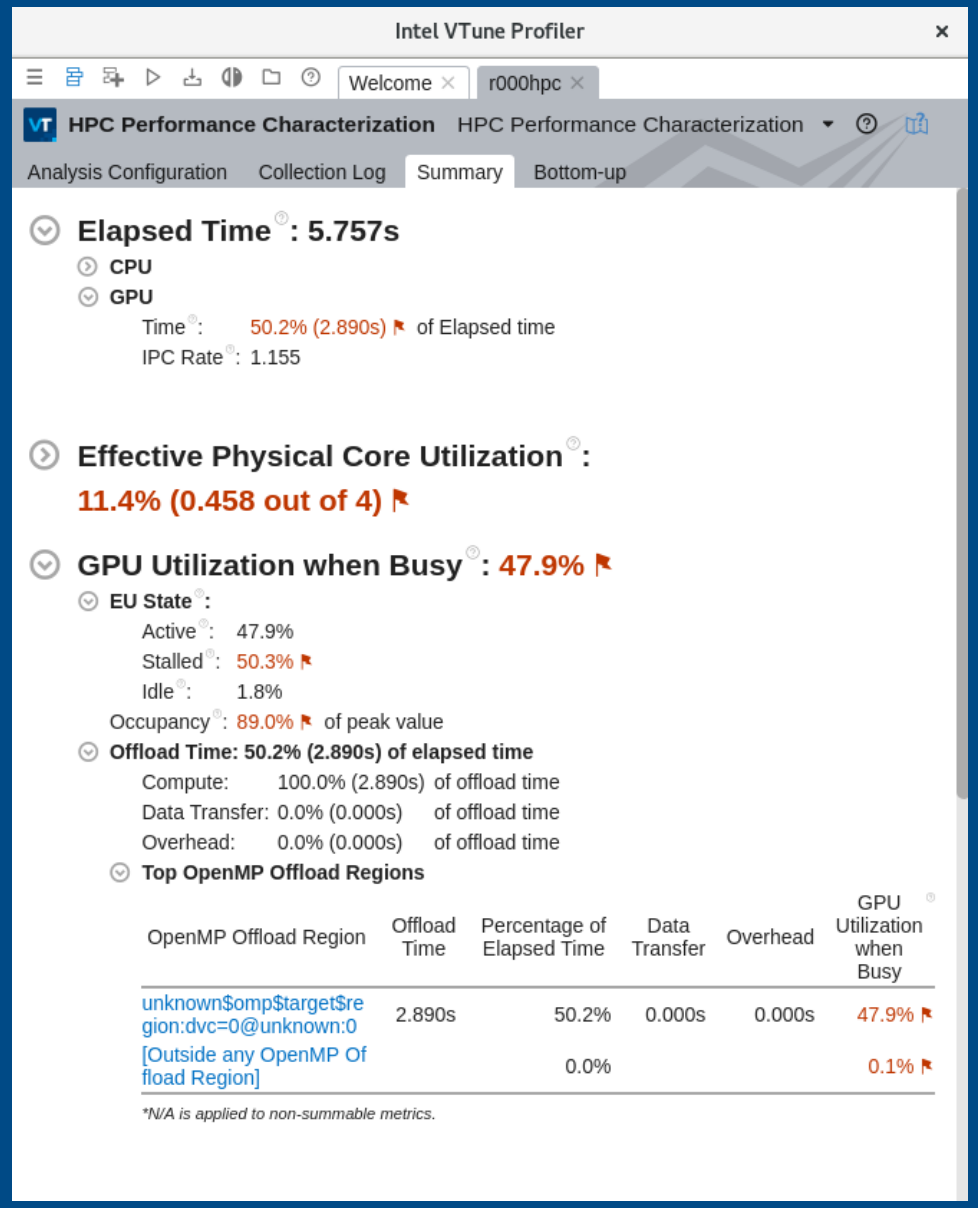
インテル® VTune™ プロファイラーの GPU オフロード機能を使用して、GPU をどの程度効率良く使用しているか確認します

インテル® VTune™ プロファイラーは、同期された CPU と GPU のタイムラインを表示します

GPU オフロードでは、GPU 実行ユニットがストールしていることが、タイムラインのバー (濃い赤) で表示されています

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するのに役立ちます
- GPU オフロード解析との主な違いは、データが OMPT インターフェイスを介して収集されることです

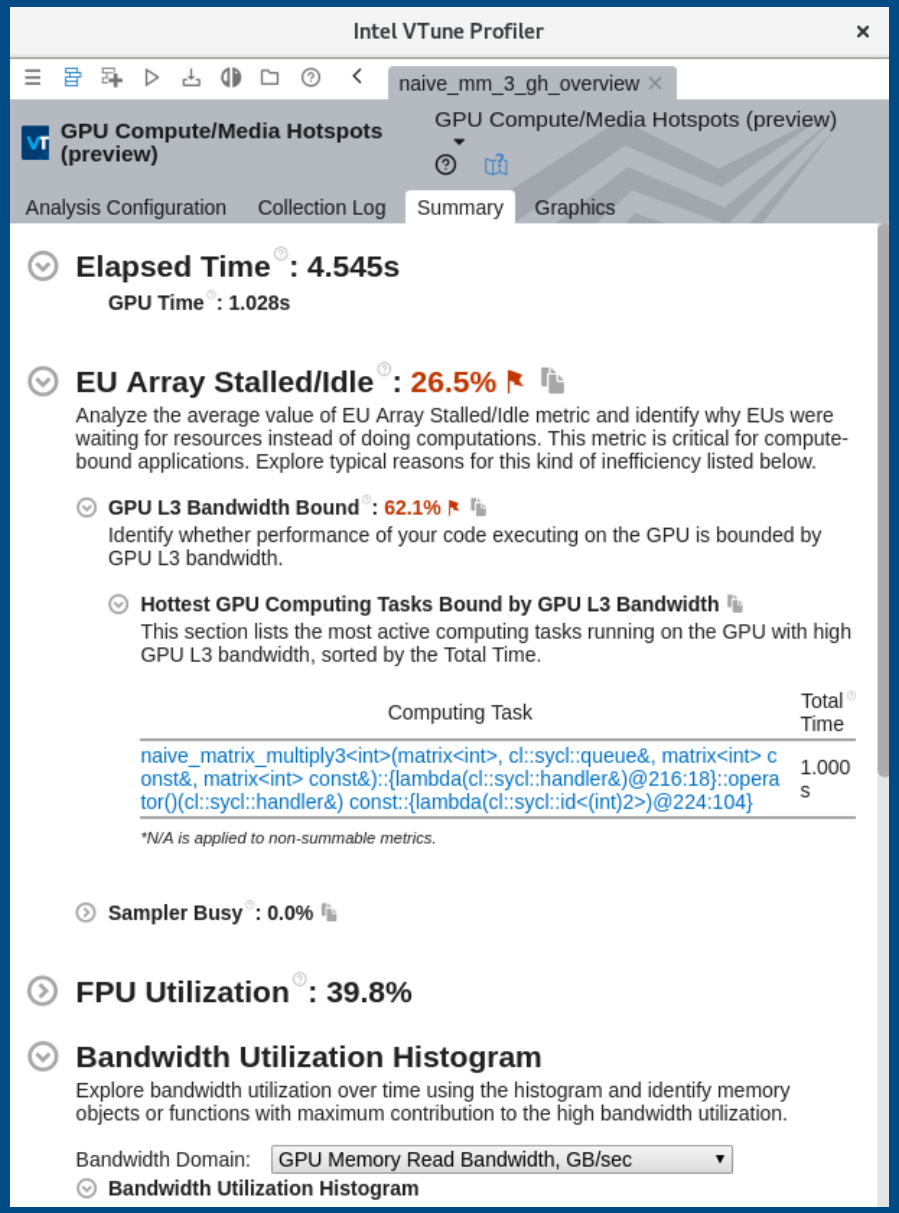


インテル® VTune™ プロファイラーのドキュメント:

HPC パフォーマンス特性解析

GPU 計算/メディア・ホットスポット

- 最も時間のかかる GPU カーネルを解析し、GPU ハードウェア・メトリックをベースに GPU の使用状況を特徴付け、メモリー・レイテンシーや非効率なカーネル・アルゴリズムによって引き起こされるパフォーマンスの問題を特定して、命令タイプごとの GPU 命令を解析します



インテル® VTune™ プロファイラーのドキュメント:

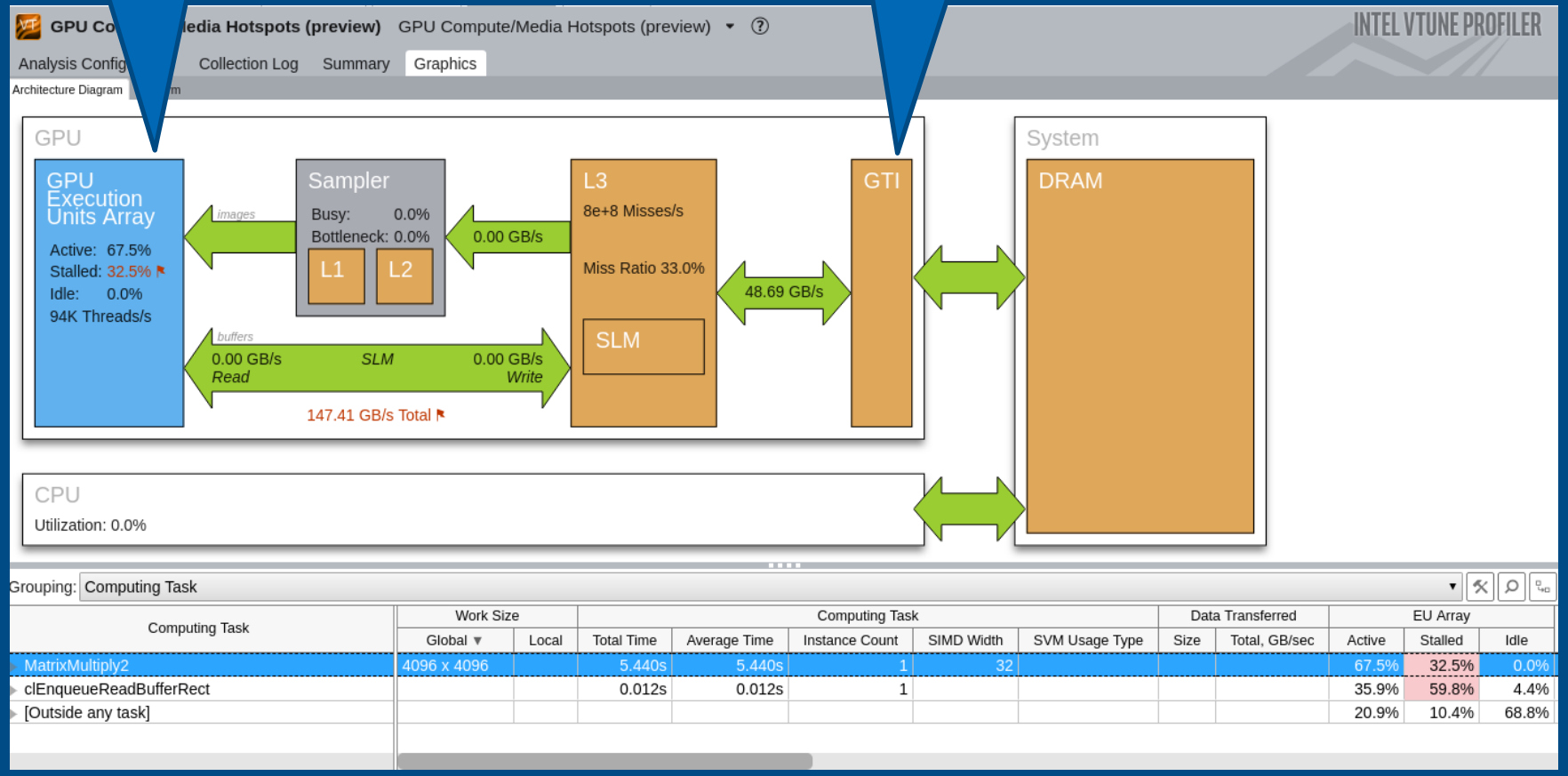
GPU 計算/メディア・ホットスポット解析

インテル® VTune™ プロファイラーを使用して GPU 使用状況を最適化

GPU 計算/メディア・ホットスポット

アクティブとアイドル
EU アクティビティー

複数レベルの帯域幅



インテル® VTune™ プロファイラーの GPU ホットスポット解析を実行して、GPU 利用率低下とストールの原因を特定します

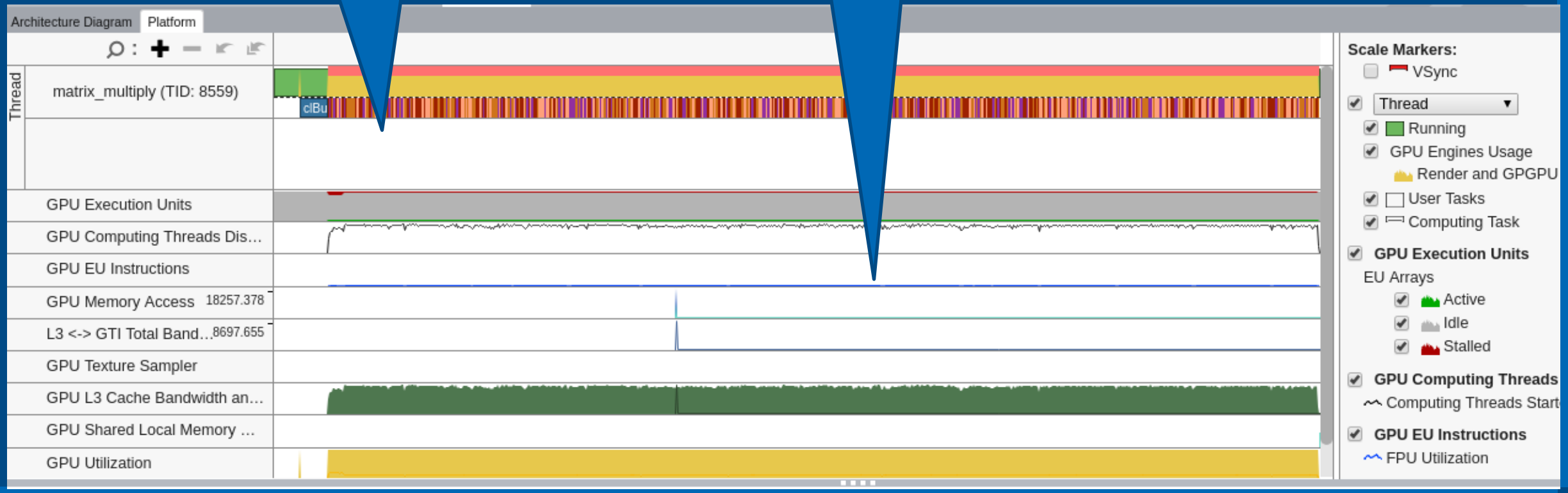
GPU ホットスポットの **[グラフィックス]** タブをクリックすると、アーキテクチャーのハイレベルな図が表示されます

インテル® VTune™ プロファイラー (インテル® GPU 用)

関係を示すタイムライン

過大または過小なカーネル･アクティビティーを特定します

GPU アクティビティーをカーネルおよびスレッドに関連付けます



インテル® VTune™ プロファイラー (DPC++ コード用)

Source | Assembly

…	Source	Computing Task: Total Time	Computing Task: Average Time	Computing Task: Instance Count	Data Transferred: Size	Data Transferred: Total, GB/sec	EU Array: Active	EU Array: Stalled	EU Array: Idle
183	hdlr.parallel_for<class MatrixMultiply3>(matrixRange, [=](dpcpp::id<2> id)	5.484s	5.484s	1			53.9%	46.1%	0.0%
184	{								
185	size_t i = id[0], j = id[1];								
186									
187	T c = {};								
188	for (size_t k = 0; k < w; k++)								
189	{								
190	c += ra[i][k] * rb[k][j];								
191	}								

DPC++ のコード行レベルでの
パフォーマンス・メトリック

Source | Assembly | Assembly grouping: Address

…	Source	Total Time
179	auto ra = bA.template get_access<dpcpp::access::mode::read>(hdl	
180	auto rb = bB.template get_access<dpcpp::access::mode::read>(hdl	
181	auto rc = bC.template get_access<dpcpp::access::mode::discard_w	
182		
183	hdlr.parallel_for<class MatrixMultiply3>(matrixRange, [=](dpcpp	5.484s
184	{	
185	size_t i = id[0], j = id[1];	
186		
187	T c = {};	
188	for (size_t k = 0; k < w; k++)	
189	{	
190	c += ra[i][k] * rb[k][j];	
191	}	
192	rc[i][j] = c;	
193	});	
194	});	

Addr…	Sour…	Assembly
0x10	183	(W) or (1|M0) cr0.0<1>:ud cr0.0<0;1,0>:ud 0x4C0:uw {Switch}
0x20	183	(W) mul (1|M0) r5.0<1>:d r14.1<0;1,0>:d r97.6<0;1,0>:d
0x30	183	(W) mul (1|M0) r23.0<1>:d r14.0<0;1,0>:d r97.1<0;1,0>:d {Compacted}
0x38		mov (16|M0) r98.0<1>:f 0x0:f
0x48		mov (16|M16) r100.0<1>:f 0x0:f
0x58	188	add (8|M0) r15.0<1>:q r3.0<8;8,1>:uw r5.0<0;1,0>:ud
0x68	188	add (8|M8) r17.0<1>:q r3.8<8;8,1>:uw r5.0<0;1,0>:ud
0x78	188	add (8|M16) r19.0<1>:q r4.0<8;8,1>:uw r5.0<0;1,0>:ud
0x88	188	add (8|M24) r21.0<1>:q r4.8<8;8,1>:uw r5.0<0;1,0>:ud
0x98	188	add (8|M0) r24.0<1>:q r1.0<8;8,1>:uw r23.0<0;1,0>:ud
0xa8	188	add (8|M8) r26.0<1>:q r1.8<8;8,1>:uw r23.0<0;1,0>:ud
0xb8	188	add (8|M16) r28.0<1>:q r2.0<8;8,1>:uw r23.0<0;1,0>:ud
0xc8	188	add (8|M0) r103.0<1>:q r15.0<4;4,1>:q r7.1<0;1,0>:ud
0xd8	188	add (8|M8) r105.0<1>:q r17.0<4;4,1>:q r7.1<0;1,0>:ud
0xe8	188	add (8|M16) r107.0<1>:q r19.0<4;4,1>:q r7.1<0;1,0>:ud
0xf8	188	add (8|M24) r109.0<1>:q r21.0<4;4,1>:q r7.1<0;1,0>:ud
0x108	188	add (8|M24) r30.0<1>:q r2.8<8;8,1>:uw r23.0<0;1,0>:ud
0x118		(W) mov (1|M0) r102.0<1>:q 0:w

計算カーネルの
GPU アセンブリー

インテル® VTune™ プロファイラーによる GPU 最適化の概要

https://www.isus.jp/products/vtune/vtune-cookbook-sw-optimization-for-gpu/

インテル® GPU を理解する

並列処理の採用:

GPU は、同時に実行可能なタスクに分割できるワークロードに適しています。GPU のシングルコアのシリアル・パフォーマンスは、CPU よりもかなり低くなります。そのため、アプリケーションは、GPU で利用可能な大規模な並列処理を活用する必要があります

データの適切な移動:

GPU の時間と空間的な局所性を活用するように、データを適切に処理します。最高のパフォーマンスを得るには、レジスターとキャッシュを利用してデータを隣接して格納することが重要です

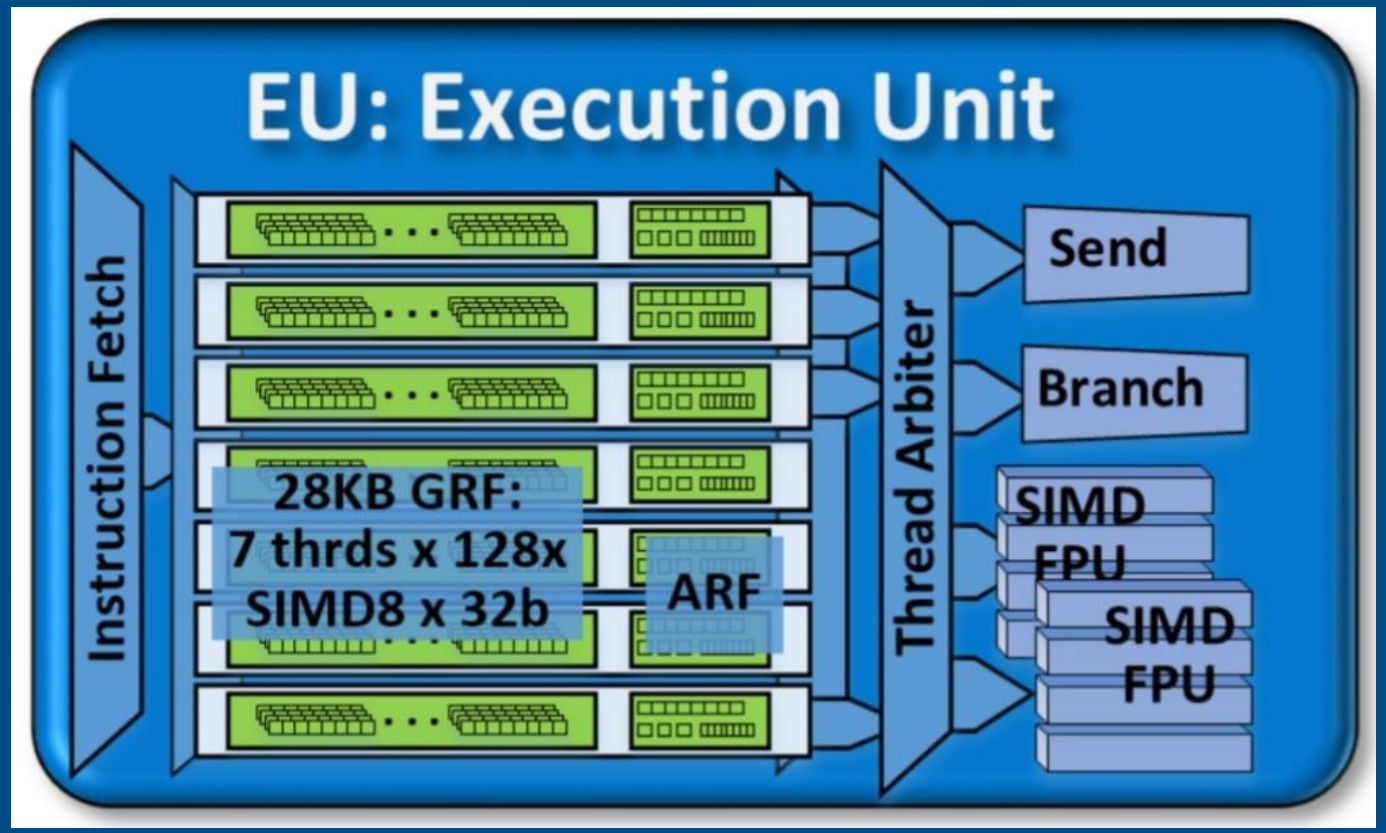
オフロードモデルの使用:

ワークロードの最も重要な部分は GPU を利用して処理されますが、ほかのワークロード・タスクを実行する CPU も重要であることに変わりはありません

計算モデルで GPU を利用するポイント

- オフロードするコード領域
- オフロードの方法
- GPGPU アルゴリズムの記述方法
- インテル® VTune™ プロファイラーの GPU オフロード解析を使用して GPU オフロードのパフォーマンスを解析する方法

インテル® GPU のアーキテクチャー



Gen9 GT2 GPU の例

EU は、同時マルチスレッディング (SMT) と細粒度のインターリーブ・マルチスレッディング (IMT) を組み合わせたものです。EU は、複数の問題、SIMD ALU (Single Instruction Multiple Data 演算ユニット) を処理するコンピューティング・プロセッサーです

これらの SIMD ALU は、複数のスレッドでパイプライン化されます。SIMD ALU は、高スループットの浮動小数点演算と整数演算に役立ちます

効率良いスケジューリングの重要性

- 通常、EU には機能処理ユニットよりも多くのハードウェア・スレッドが供給されます。オーバーサブスクリプションが発生する可能性がありますが、同時に待機中のデータによるストールを隠蔽するのにも役立ちます
- この方法によるスレッドのスケジューリングにはコストがかかるり、スケジューリングを効率良く費用効果を高くするには、すべての EU をできるだけビジーにすることが重要です。次のケースでは、スケジューリングの効果は低くなります
  - 計算量が少なすぎる。スケジューリングのオーバーヘッドが、有用な計算を完了するために費やされる時間に匹敵する可能性があります
  - 計算量が多すぎる。スレッド間のワークの分散が不均等になり、GPU のすべての EU の全体的な占有率が低くなる可能性があります

浮動小数点ユニットを使用した SIMD 実行

EU のプライマリー計算ユニットは SIMD 浮動小数点ユニット (FPU) のペアです。これらの FPU は、実際には浮動小数点と整数の両方の計算をサポートしています。次の表は、これらの FPU の SIMD 実行機能を説明したものです

データサイズ	データ型	SIMD 操作の数
16 ビット	整数	8
16 ビット	浮動小数点	8
32 ビット	整数	4
32 ビット	浮動小数点	4

浮動小数点ユニットを使用した SIMD 実行 (続き)

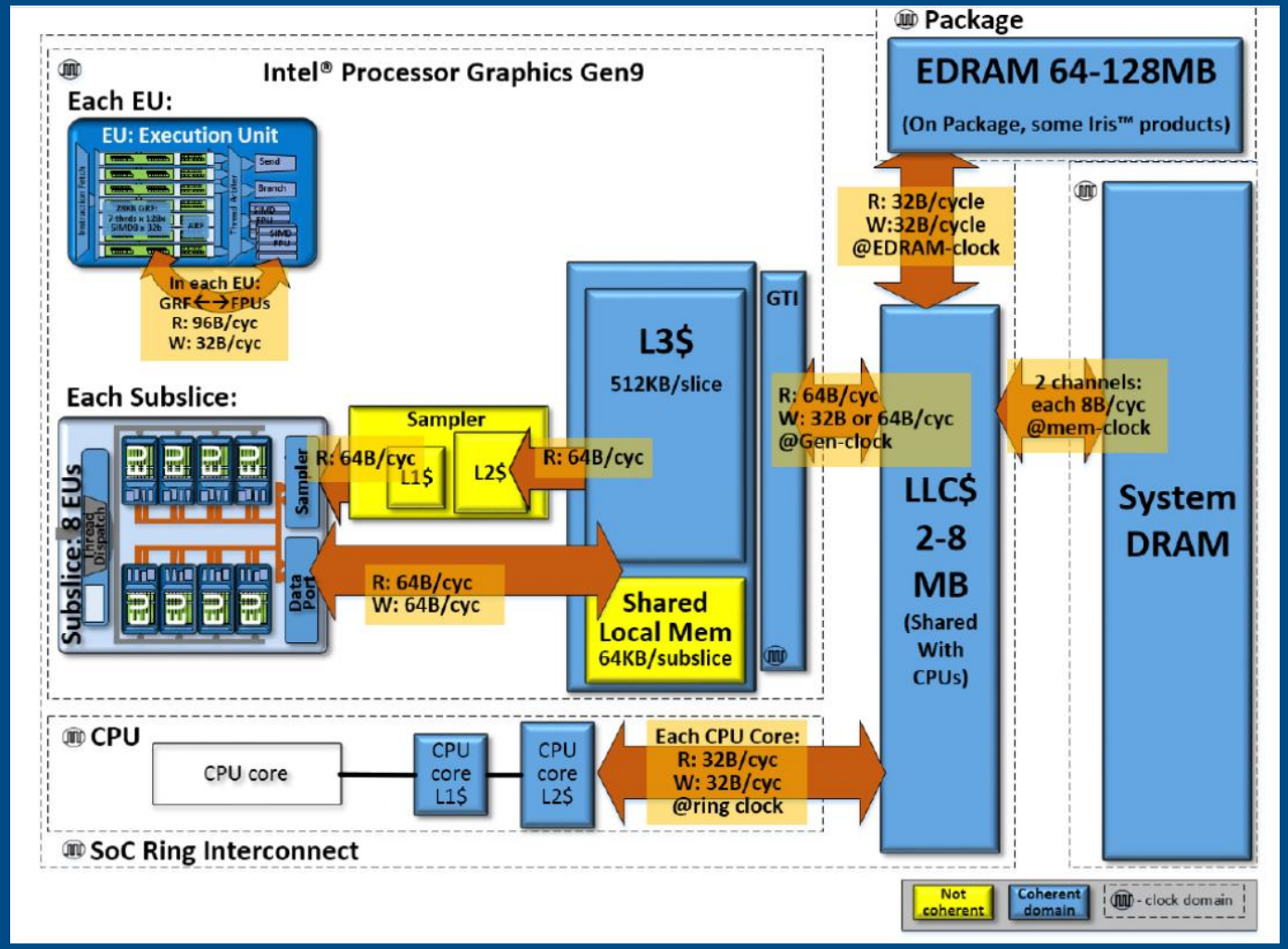
EU IPC Rate (EU IPC レート) メトリックは、FPU の飽和状態を示す指標です。たとえば、2 つの非ストールスレッドでマシンの浮動小数点計算スループットが飽和する場合、メトリックは 2 です。通常、このメトリックは理論上の最大値である 2 よりも低くなります

SIMD 幅の値	意味
4 未満	コンパイラーがループのベクトル化を妨げている原因を確認します。
4 以上	コンパイラーによる命令のベクトル化は成功しています。データの依存関係を削除するか、コードにループアンロール手法を適用すると、この値をデータの局所性とキャッシュの再利用に適した 16 または 32 に増やすことができます。

FPU が飽和しておりデータ幅が狭い場合、命令レベルの並列処理は不適切であると言えます。この場合、**SIMD Width (SIMD 幅)** メトリックを確認します

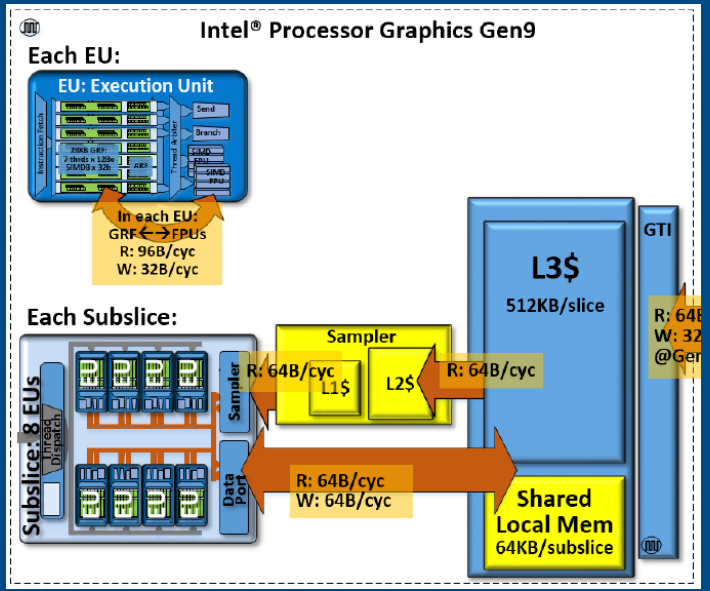
メモリー・サブシステム

- Gen9 GT2 GPU には、**UMA (Unified Memory Architecture)** を備えた独自のメモリー・サブシステムが用意されています
- このメモリー・サブシステムは、物理メモリーを CPU と共有して、ゼロコピーバッファー転送を効率良く行います
- この機能により、CPU と GPU 間のデータ転送が高速化されます



メモリー・サブシステム (続き)

Gen9 メモリー・アーキテクチャーでは、各スライスはそのスライスの L3 キャッシュにアクセスできます。スライスはサブスライスで構成されます。各サブスライスには、次のものが含まれます



- ローカル・スレッド・ディスパッチャー
- 命令キャッシュ
- L3 へのデータポート
- 共有ローカルメモリー (SLM)

次のいずれかの方法で、データの局所性を制御できます

- 特定のデータアクセス。L3 キャッシュにデータを格納するハードウェアを支援します
- ワークグループでアクセス可能なローカルメモリーを割り当てる特別な API。ハードウェアにより SLM で提供されます

インテル® VTune™ プロファイラーの GPU プロファイル機能を利用

GPU 解析をサポートするように調整されたインテル® VTune™ プロファイラーの主要な機能を取り上げます。次のワークフローは、これらの機能の説明です

1. アプリケーションの GPU オフロード解析を実行します
2. GPU 計算/メディア・ホットスポットのプロファイルを収集します
3. メモリー階層ダイアグラムを使用して、最も効率の悪い計算タスクに取り組みます
4. カーネルで [Instructions Count] プリセット解析を実行します
5. 大規模な計算カーネルには、GPU 計算/メディア・ホットスポット解析の [Basic Block Latency] プリセットを使用します
6. [Memory Latency] プリセットを使用して、重大な実行ストールを示したメモリー・アクセス・コードを調べます
7. 目標のパフォーマンス・メトリックになるまで、改善したアルゴリズムで GPU 計算/メディア・ホットスポット解析を繰り返します

インテル® GPU にオフロードする場合の最適化手法

ヘテロジニアス・アプリケーションは通常、アクセラレーターにオフロードするコード領域を特定して設計されます。オフロードするコード領域を特定できない場合は、インテル® Advisor のオフロードのモデル化 (英語) を使用して特定します

ここでは GPU にオフロードするコードをすでに特定していると仮定しています。次に、ホスト側でオフロードを実装する最適な方法を示します

ステップ 1: デバイスの使用率を調べる

ステップ 2: GPU でのコード実行の効率を定義する

ステップ 3: GPU オフロード解析を実行する

ステップ 4: GPU 計算/メディア・ホットスポット解析を実行する

ステップ 5: 追加のカーネルコードの最適化

インテル® VTune™ プロファイラーによる解析例

行列サンプル･アプリケーション の例

```
for (size_t i = 0; i < w; i++)
     for (size_t j = 0; j < w; j++) {
         c[i][j] = T{};
             for (size_t k = 0; k < w; k++)
                 c[i][j] += a[i][k] * b[k][j];
     }
```

アクセラレーターにオフロードするコード領域を特定します。通常は、最も外側のループが適切な候補です。しかし、この例では、最も内側のループが計算カーネルである可能性があります

GPU にオフロードするホットな計算タスクを特定

GPU Offload　GPU Offload　INTEL VTUNE PROFILER

Analysis Configuration　Collection Log　Summary　Graphics　Platform

Recommendations

GPU Utilization:17.9%
GPU utilization is low. Switch to the Bottom-up view for in-depth analysis of host activity. Poor GPU utilization can prevent the application from offloading effectively.

EU Array Stalled/Idle:97.3%
GPU metrics detect some kernel issues. Use GPU Compute/Media Hotspots (preview) to understand how well your application runs on the specified hardware.

Elapsed Time: 123.223s

GPU Utilization: 17.9%
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 Utilization
GPU Utilization break

GPU Engine / Packet
Render and GPGPU
Unknown
*N/A is applied to non-sum

Grouping: Computing Task

Computing Task	Computing Task					Work Size		EU Array			EU Threads Occupancy
	Total Time	Average Time	Instance Count	SIMD Width	SVM U...	Global	Local	Active	Stalled	Idle	
matrixMultiply1<float,	0.354s	0.000s	65,536	1		256		0.4%	6.9%	92.6%	1.4%
clEnqueueReadBuffer	2.517s	0.000s	65,536					29.8%	8.8%	61.4%	25.2%
[Outside any task]	0s							1.6%	0.6%	97.8%	1.3%

5.2s　5.4s　5.6s　5.8s　6s　6.2s　6.4s　6.6s

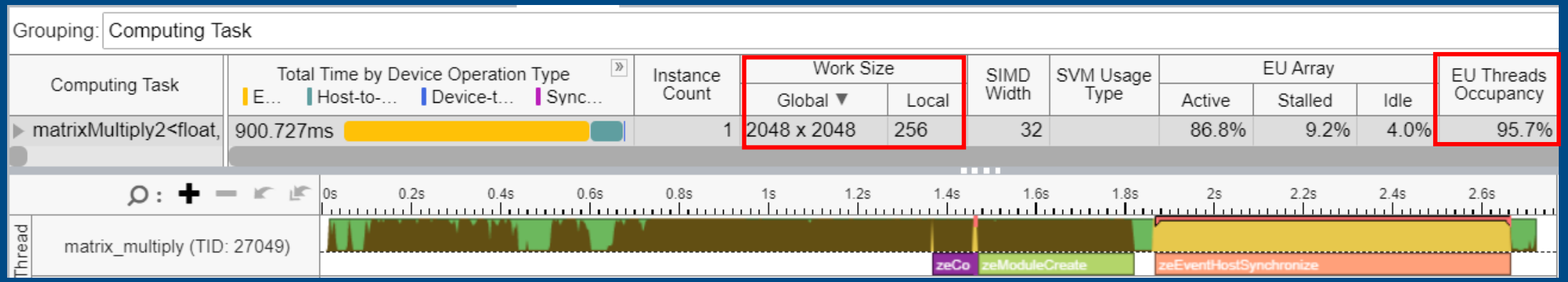
Thread　mainCRTStartup (TID: 3912)　clLinkProgram

Scale Markers:
VSync
Thread

推奨事項に従って、まず低い GPU 利用率の原因となる可能性のあるホスト・アクティビティーを調べます

ワークの分散が非効率

外側のループをオフロード



行列のサイズは 2048 X 2048 に増加し、ウォールクロック･パフォーマンスは 10 倍以上高速でした。**[EU Threads Occupancy (EU スレッドの占有期間)]** メトリックは 95.7% と高い値です。これは、実行ユニットで利用できる十分なワークがあることを示しています

デバイスの操作によるタスク時間の特徴付け

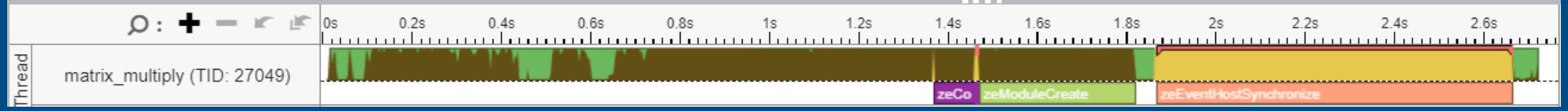
Grouping: Computing Task

Computing Task	Total Time by Device Operation Type				Instance Count	Work Size		SIMD Width
	Execution	Host-to-Device Transfer	Device-to-Host Transfer	Synchronization		Global ▼	Local	
matrixMultiply2<float,	796.320ms	101.680ms	2.728ms	0ms	1	2048 x 2048	256	32

タイムラインを見ると、データ転送が 100 ミリ秒しかかかっていないのに対して、実行に 800 ミリ秒近くかかっている計算タスクが分かります。このデータ実行とデータ転送の比率は、アルゴリズムを改良することで改善できます

Grouping: Computing Task

Computing Task	Total Time by Device Operation Type (E… / Host-to-… / Device-t… / Sync…)	Instance Count	Work Size		SIMD Width	SVM Usage Type	EU Array			EU Threads Occupancy
			Global ▼	Local			Active	Stalled	Idle	
matrixMultiply2<float,	900.727ms	1	2048 x 2048	256	32		86.8%	9.2%	4.0%	95.7%



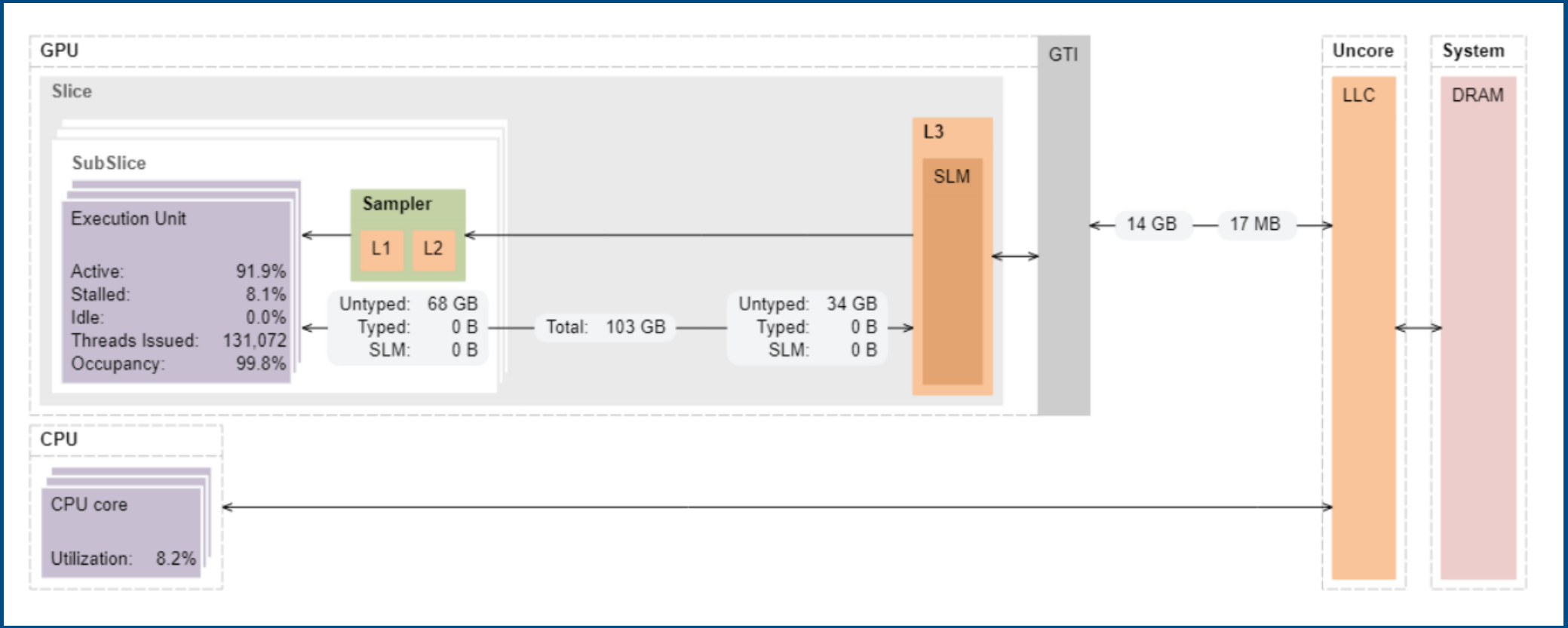
カーネルが計算依存なのかメモリー依存なのかを特定

Computing Task	EU Instructions			L3 Bandwidth, GB/sec	Untyped Memory Bandwidth, GB/sec		Shared Local Mem...	
	IPC Rate	2 FPUs active	Send active		Read	Write	Read	Write
matrixMultiply2<float,	1.705	63.5%	20.0%	132.179	88.105	44.074	0.000	0.000

GPU ホットスポット解析には、いくつかの事前定義済みプロファイルやプリセットが用意されています。これらのプリセットを使用して、メモリーアクセスや計算効率に関連する異なるメトリックを収集できます

カーネル実行を適切に理解できるように、**[Full Compute (完全な計算)]** プリセットを使用しました。このプリセットの情報から、カーネルの実行で、EU FPU が 63.5% の時間しかアクティブでなかったことが分かります

GPU メモリー・サブシステムのカーネルデータ転送

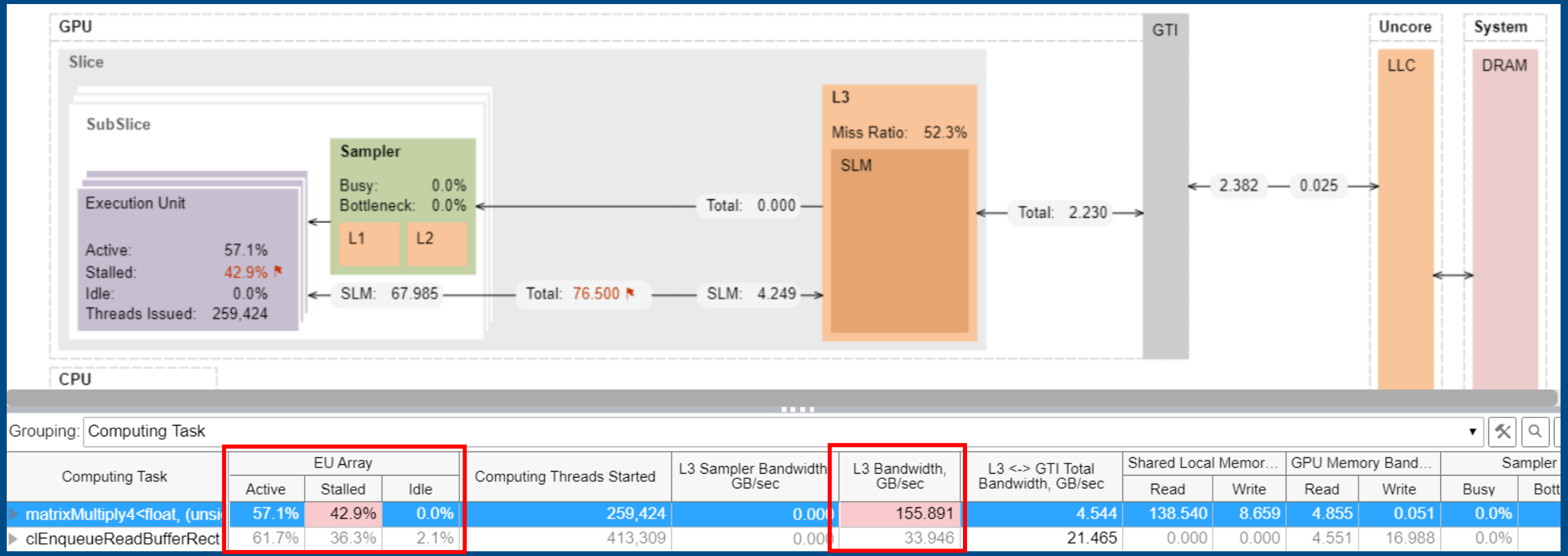


タイル化された乗算の実装

```
i, j // グローバル･インデックス
for (size_t tidx = 0; tidx < TILE_COUNT; tidx++)
     ti, tj // ローカル･インデックス
     ai, aj, bi, bj // global to local idx
     ta[ti, tj] = a[ai, aj]
     tb[ti, tj] = b[ai, aj]

     for (size_t tk = 0; tk < TILE_SIZE; tk++)
            c[i][j] += ta[ti][tk] * tb[tk][tj];
 }
```

タイル化されたカーネルデータ転送 (SLM)



Computing Task	EU Array			Computing Threads Started	L3 Sampler Bandwidth GB/sec	L3 Bandwidth, GB/sec	L3 <-> GTI Total Bandwidth, GB/sec	Shared Local Memor...		GPU Memory Band...		Sampler	
	Active	Stalled	Idle					Read	Write	Read	Write	Busy	Bott
matrixMultiply4<float, (unsi	57.1%	42.9%	0.0%	259,424	0.000	155.891	4.544	138.540	8.659	4.855	0.051	0.0%	
clEnqueueReadBufferRect	61.7%	36.3%	2.1%	413,309	0.000	33.946	21.465	0.000	0.000	4.551	16.988	0.0%	

Grouping: Computing Task

Computing Task	Work Size		Computing Task				
	Global	Local	Total Time ▼	Average Time	Instance Count	SIMD Width	SVM Usag...
matrixMultiply4<float, (unsig	2048 x 2048	16 x 16	490.727ms	490.727ms	1	16	
clEnqueueReadBufferRect			3.116ms	3.116ms	1		
[Outside any task]			0ms	0ms	0		

改善の余地はあるか：理論上の最小時間を計算

Gen9 GT2 GPU アーキテクチャーのパラメーターから、この GPU には 24 の EU が含まれていることが分かります

各 EU には 2 つの FPU (SIMD-4) があります。各 FPU は 2 つの演算 (乗算+加算) を実行できます。最大コア周波数は 1.20GHz で、最大 FP パフォーマンスは次のとおりです

24 * 2 * 4 * 2 = 384 FLOP/サイクル (32 ビット浮動小数点)

384 * 1.2 = 460.8 GFLOPS

行列乗算実装の FP 演算の数は $2*N^3$ (N=2048 の場合、約 17.2 GOPS)

理論的には、データアクセスの非効率性および帯域幅の制約による制限がなかった場合、アルゴリズムは 17.2 / 460.8 = 0.037 秒 (37 ミリ秒) で計算できます

カーネルで実行された最良の時間は 490 ミリ秒であり、理論上の計算時間よりも 10 倍以上遅くなっています

つまり、パフォーマンスを向上する余地はまだあります

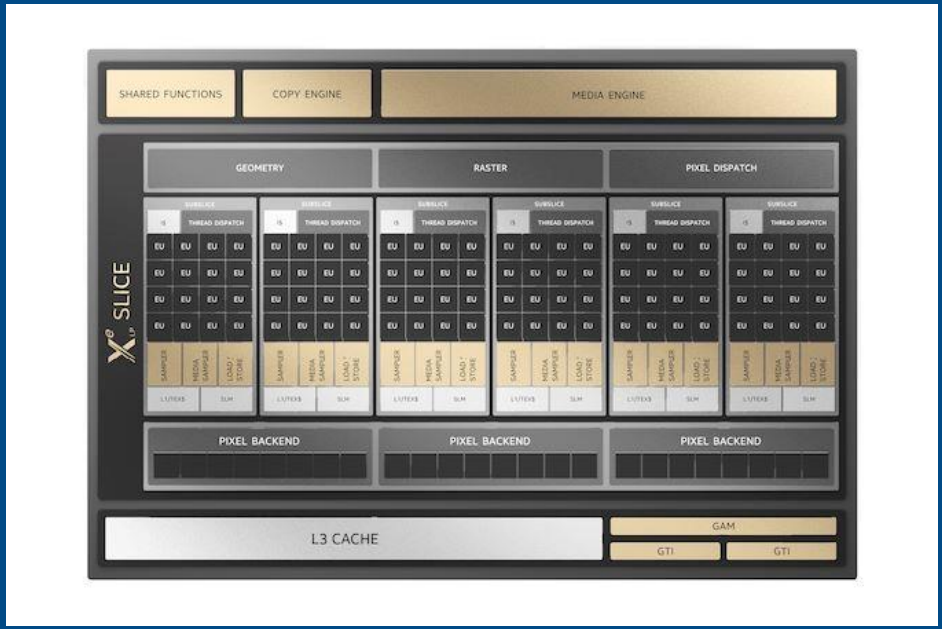
スケーリング・パフォーマンス

行列乗算サンプルのような高度に並列化されたアプリケーションは、GPU リソースが増加することにより効率が向上します

Gen9 シリーズの GPU には、48 の EU を含む GT3 と 72 の EU を含む GT4 があります

しかし、組込み型の GPU には基本的な領域の制限があり、スケーリングを高めるために EU を追加したり、データアクセスを高速化するためにキャッシュブロックを大きくしたりすることはできません

ディスクリート GPU では、領域や電力の制約による制限は少なくなります。システムで 1 つまたは複数の GPU との統合が許可されている場合、アクセラレーターのパフォーマンスをスケールアップできます



インテル® Iris® Xe MAX マイクロアーキテクチャーの高レベルのビュー

インテル® Iris® Xe MAX GPU の例

タイル化されたカーネルのタイミング

Grouping: Computing Task

Computing Task	Work Size		Computing Task				
	Global	Local	Total Time ▼	Average Time	Instance Count	SIMD Width	SVM Usag...
matrixMultiply4<float, (unsig	2048 x 2048	16 x 16	490.727ms	490.727ms	1	16	
clEnqueueReadBufferRect			3.116ms	3.116ms	1		
[Outside any task]			0ms	0ms	0		

Gen9

Computing Task	Work Size		Computing Task			
	Global ▼	Local	Total Time	Average Time	Instance Count	SIMD Width
matrixMultiply4<	2048 x 2048	16 x 16	111.455ms	111.455ms	1	16
clEnqueueRead			159.791ms	159.791ms	1	
[Outside any tas			0ms			

インテル® Iris® Xe MAX

- カーネル実行は約 4 倍高速化されました

Gen9 GPU の 24 の EU に対してインテル® Iris® Xe MAX GPU には 96 の EU が含まれていることから、これは予想どおりの結果と言えます

改善の余地はあるか

次の表を見ると、EU が実行中に 51% の時間でまだストールしていることが分かります。これは、(一般的な行列アルゴリズムでよく知られている) メモリーからのデータの待機が原因であると考えられます。具体的な原因を調べます

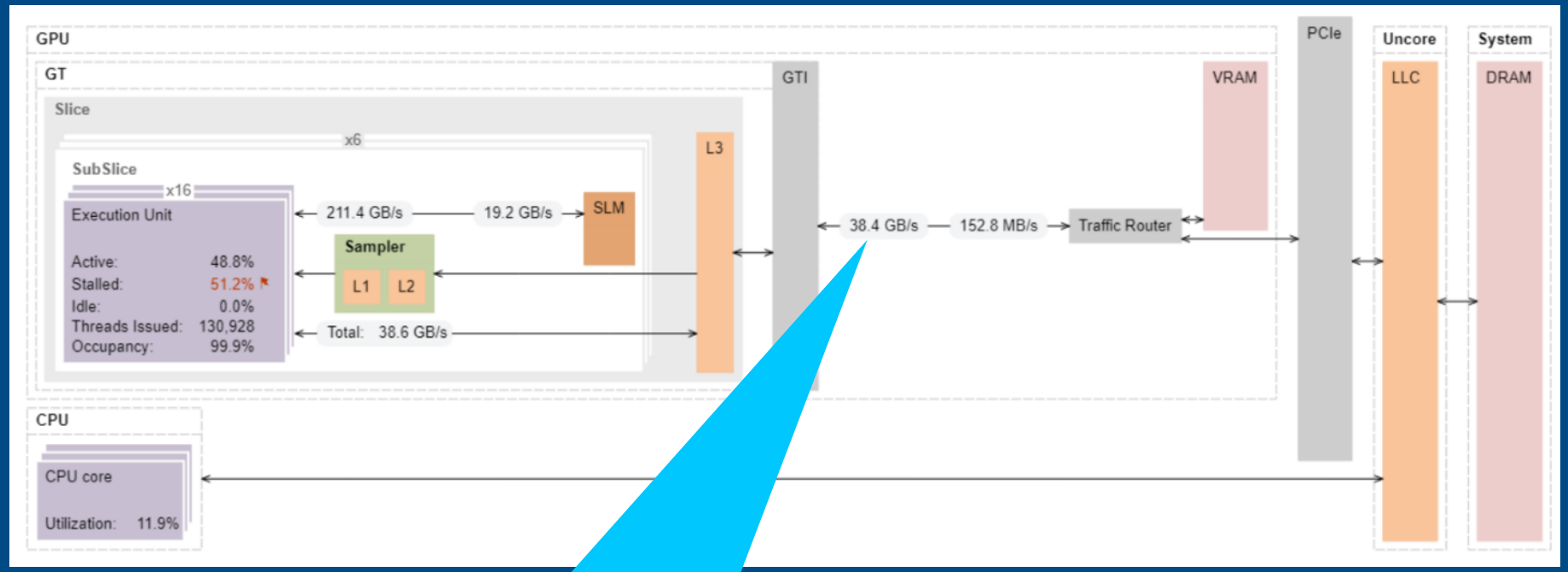
Computing Task	EU Array			EU Threads Occupancy	Computing Threads Started	L3 Bandwidth, GB/sec	Shared Local Memory Bandwidth, GB/sec		GPU Memory Bandwidth, GB/sec	
	Active	Stalled	Idle				Read	Write	Read	Write
matrixMultiply4<float,	48.8%	51.2%	0.0%	99.9%	130,928	6.5%	17.8%	19.220	19.4%	0.153
clEnqueueReadBuffer	0.3%	99.6%	0.0%	94.7%	261,730	0.2%	0.0%	0.033	0.2%	0.419
[Outside any task]	0.0%	2.7%	97.3%	2.5%	49,710	0.0%	0.0%	0.000	0.0%	0.174

タイル化されたカーネルの EU Array (EU アレイ) メトリック

最大帯域幅の割合を見ると、L3 と GPU メモリーの帯域幅は最大値からかなり離れているため、ボトルネックではないことが分かります

インテル® Iris® Xe MAX マイクロアーキテクチャー
メモリー階層ダイアグラムで全体像を把握

行列 *a* と行列 *b* のデータが PCIe* 経由で GTI に転送されます



測定されたデータ読み取り速度は GTI インターフェイスで 38GB/秒

PCIe* 3.0 ×16 の理論上の最大値は片方向で 16GB/秒であるため、PCIe* の帯域幅に制限されていることが分かります

最適化のヒント

- すべてのデータは VRAM と EU ストールからフェッチされています。これは、ビデオメモリーから EU へのデータ移動のレイテンシーで定義されます

- EU と L3 間のデータ・トラフィックは GTI と外部トラフィック・ルーター間のデータ・トラフィックと同じであるため、L3 キャッシュを適切に再利用することにより、パフォーマンスをさらに最適化できます

  例えば、各 GPU スライスの L3 キャッシュに収まるブロックサイズで、セカンドレベルの行列タイリングを行います

まとめ

一般に、ヘテロジニアス・アプリケーションでは、特定のワークロードをアクセラレーターにオフロードする場合、GPU などの超並列アクセラレーター・マシンに十分な計算タスクを提供することが不可欠です

- オフロードしたタスクのデータ転送とタスク・スケジューリングのオーバーヘッドを予測することにより、GPU の効率を向上します
- インテル® VTune™ プロファイラーの GPU オフロード解析の **[GPU Utilization (GPU 利用率)]** メトリックおよび **[GPU Occupancy (GPU 占有率)]** メトリックを使用して、GPU 使用の非効率性を予測します
- 計算タスク実行のパフォーマンスは、実行ユニットの不足、メモリー・サブシステムまたはインターフェイスのボトルネックの存在など、いくつかのマイクロアーキテクチャーの要因によって制限される場合があります

  GPU 計算/メディア・ホットスポット解析を実行して、これらの制限を特定します。GPU メモリー階層ダイアグラムのボトルネックと、すべての計算タスクの詳細なマイクロアーキテクチャーのメトリックをハイライトします。複雑なカーネルの場合、レイテンシー解析を使用して、カーネル内の最も重要なコードを特定します

CPU バージョンの最適化について

https://www.isus.jp/products/oneapi/webinar/

オンデマンド	インテル® VTune™ プロファイラー入門: サンプルコードを使用してツールの使い方を理解する	インテル® VTune™ プロファイラー 2020 に含まれるサンプルコードを使用して、プログラムの最適化ステップでどのようにプロファイラーを活用できるか、収集されたパフォーマンス・データの解釈を含めて解説します。 このセッションは 2 つのパートに分かれています。パート 1 では、同一プラットフォームの最適化ステップでパフォーマンス・データがどのように変化するかを説明します。パート 2 では、同一プログラムを異なるプラットフォームで実行した際にパフォーマンス・データがどのように変化するかを説明します。	初心者向け	PDF 資料 / ウェビナー動画 (パート 1) / ウェビナー動画 (パート 2)

著作権と商標についておよび最適化に関する注意事項

- 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® SSE2)、インテル® ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。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®