インテル® oneAPI ツールキット 2023 によって異種デバイスを活用する

ハイパフォーマンス・ソフトウェア・カンファレンス 2023

エクセルソフト株式会社

ご紹介したいこと

- 現行の最新バージョン 2023 を使った DPC++ (C++ with SYCL*) および Fortran 言語向けに GPU の利用方法と、そのプログラムの解析手段をご紹介
  - インテル® oneAPI ツールキット概要
  - DPC++ および Fortran における異種デバイスの利用
  - 解析ツールによる GPU を扱うプログラムの解析

インテルの開発ツールと ハードウェア・プラットフォーム

HPC & データセンター	AI & ビジュアライゼーション	組込みシステム & IoT

パフォーマンス	生産性	ベンダー依存からの解放
最新のインテルの CPU、GPU、FPGA で計算パフォーマンスを最適化 内蔵アクセラレーターを最大限に活用 さまざまな AI フレームワークを高速化	使い慣れた言語と標準規格 レガシーコードと簡単に統合 CUDA* から SYCL* へ簡単に移行 最小限のコード変更	ベンダー固有に代わる、オープンな選択肢 アーキテクチャー変更が容易 将来のハードウェアに対応したコード

パフォーマンス・ライブラリーの利用

パフォーマンス・ライブラリー

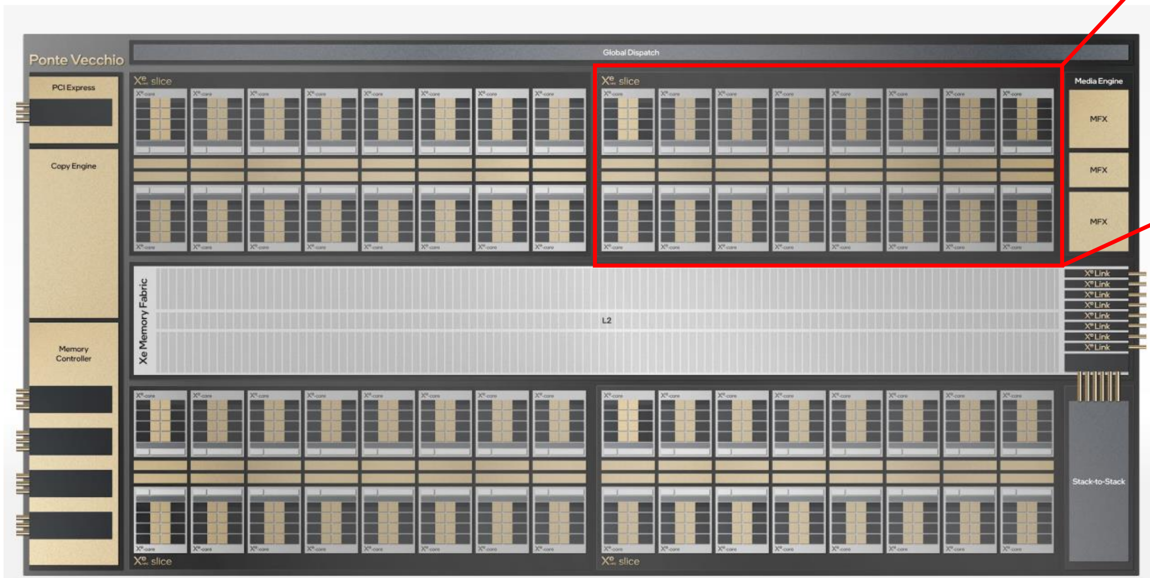
- スレッド、オフロード、数学、データ・アナリティクス、データ処理レンダリング、レイトレーシング、DNN、通信、暗号、など
- ハードウェアの価値を最大限に引き出す
- 主要なドメイン固有の関数を高速化するように設計

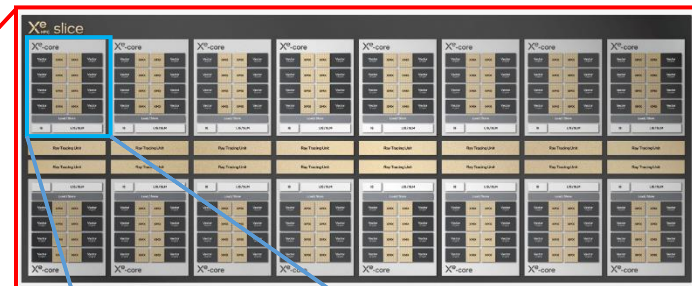
最高のパフォーマンスが得られるように各プラットフォーム向けに事前に最適化

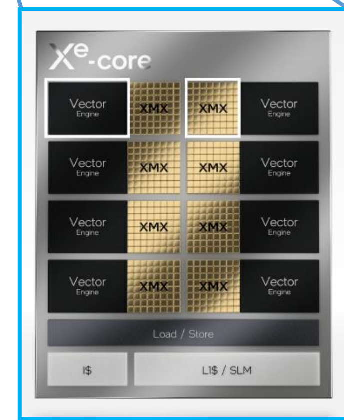
インテル® oneAPI マス・カーネル・ライブラリー **(インテル® oneMKL)**	インテル® oneAPI ディープ・ニューラル・ネットワーク・ライブラリー **(インテル® oneDNN)**
インテル® oneAPI ビデオ・プロセシング・ライブラリー **(インテル® oneVPL)**	インテル® oneAPI データ・アナリティクス・ライブラリー **(インテル® oneDAL)**
インテル® oneAPI スレッディング・ビルディング・ブロック **(インテル® oneTBB)**	インテル® oneAPI コレクティブ・コミュニケーション・ライブラリー **(インテル® oneCCL)**
インテル® oneAPI DPC++ ライブラリー **(インテル® oneDPL)**	インテル® oneAPI レンダリング・ツールキット レンダリングとレイトレーシング・ライブラリー
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® MPI ライブラリー

多くの機能を備えた開発ライブラリーを提供

GPU 構造

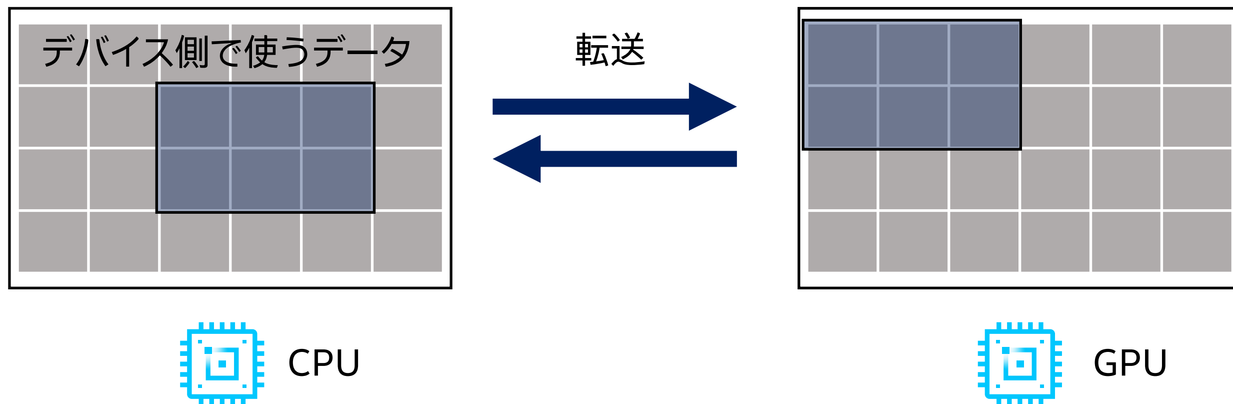






CPU と GPU

- ホストとデバイスは異なるメモリー空間を持ちます
  - デバイスからアクセスするデータは使用前にデバイス側へ転送します
  - デバイスで実行された結果をホストに転送します



生産性とパフォーマンスに優れた SYCL* コンパイラー

インテル® oneAPI DPC++/C++ コンパイラー

CPU とアクセラレーターに妥協のない並列プログラミングの生産性とパフォーマンスを提供

- ターゲット・ハードウェア間でコードの再利用が可能、特定のアクセラレーター向けのカスタム・チューニングを行うことが可能
- 単一アーキテクチャー専用の言語に代わる、オープンな業界全体の代替手段

Khronos SYCL* 標準

- C++ の生産性の利点を提供、一般的で使い慣れた C および C++ 構造を使用
- データ並列処理とヘテロジニアス・プログラミングをサポートするため Khronos Group が作成

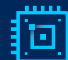
インテルの数十年にわたるアーキテクチャーとハイパフォーマンス・コンパイラーの経験を基に構築

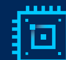
インテル® oneAPI DPC++/C++ コンパイラーとランタイム

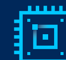
C++ with SYCL* ソースコード

Clang/LLVM

DPC++ ランタイム







DPC++ (C++ with SYCL*)

- ホスト + アクセラレーター向けのコードを単一ソースで記述します

構造	目的
キュー	ターゲットでのワーク
バッファー	データ管理
parallel_for	並列処理

- デバイスセレクターによる制御

主なデバイスセレクター	説明
default_selector_v	有効なデバイスを選択
gpu_selector_v	GPU の選択
cpu_selector_v	CPU の選択

```
#include <CL/sycl.hpp>
#include <iostream>
constexpr int num=16;
using namespace cl::sycl;

int main() {
  auto R = range<1>{ num };
  buffer<int> A{ R };

  queue{}.submit([&](handler& h) {
    auto out =
      A.get_access<access::mode::write>(h);
    h.parallel_for(R, [=](id<1> idx) {
      out[idx] = idx[0]; }); });

  auto result =
    A.get_access<access::mode::read>();
  for (int i=0; i<num; ++i)
    std::cout << result[i] << "¥n";

  return 0;
}
```

ホストコード（int main() から submit の前まで）
GPU コード（parallel_for の本体）
ホストコード（result 以降）

デバイスセレクター

- SYCL* 2020 の対応により従来のデバイスセレクターは非推奨
- SYCL* 1.2.1 で使われていたデバイスセレクターはコンパイル時に警告を出力

各デバイス向けセレクター
default_selector_v
gpu_selector_v
accelerator_selector_v
cpu_selector_v
host_selector_v

```
// SYCL* 2020
sycl::device my_gpu { sycl::gpu_selector_v };
sycl::queue my_accelerator { sycl::accelerator_selector_v };
// SYCL* 1.2.1 コードの移植のため引き続き利用可能
sycl::queue my_old_style_gpu { sycl::gpu_selector {} };
```

Codeplay* コンパイラー・プラグイン

- インテル® oneAPI ベース・ツールキットに
  NVIDIA* GPU と AMD* GPU サポートを追加
  - インテル® DPC++/C++ コンパイラーへの
    バイナリープラグインを Codeplay からダウンロード
- プラグインの対応
  - NVIDIA* GPU - DPC++ へ CUDA* バックエンドを追加
  - AMD* GPU (ベータ版) – DPC++ へ HIP バックエンドを追加
- 日本語の利用ガイド
  - oneAPI for NVIDIA* GPU 2023.1.0 ガイド
  - ベータ版 oneAPI for AMD* GPU 2023.1.0 ガイド

ifort と ifx
2 種類の Fortran コンパイラー

インテル® oneAPI ツールキットから利用できる Fortran コンパイラー

- **ifort - インテル® Fortran コンパイラー·クラシック**
  インテル® Parallel Studio XE から引き続き提供されるコンパイラー
  ベクトル化、スレッド化を支援する CPU 向け最適化を含む
- **ifx - インテル® Fortran コンパイラー**
  新しく実装された Fortran コンパイラー
  ifort と同等の最適化機能および GPU 向けのオフロード機能を提供

今回使うコンパイラー

XLSOFT

ifort の継続と今後

- ifx への移行を推奨しつつも引き続き使用可能

  CPU ターゲットのコードを保守

- ifx が ifort を置き換えできると判断

  次期バージョンに向けた最適化性能の向上

ifx への移行に向けたポーティング・ガイドを公開

ifx の主要なアップデート

- Fortran 、OpenMP* 標準のサポートの追加
  Fortran 2018 + 標準とそれ以前のすべての標準
  OpenMP* 5.x (オフロード機能含む)
- IFORT と同じ拡張機能、ディレクティブ、動作に対応
  自動並列化機能を除く
- DO CONCURRENT による GPU オフロード機能
  コンパイラー・オプションによる制御
- iso_c_binding による C/C++ との相互運用性の拡張 (Fortran 2018)

DO CONCURRENT による並列化

- DO ループの反復間にデータの依存関係がないことを指定

```
INTEGER,DIMENSION(N) :: J, K
INTEGER              :: I, M
M = 10
I = 15
DO CONCURRENT (I = 1:N, J(I)> 0)LOCAL (M) SHARED (J, K)
   M =  MOD (K(I), J(I))
   K(I) = K(I) - M
END DO
PRINT *, I, M
```

```
Linux*:  > ifx -fopenmp-target-do-concurrent -fiopenmp -fopenmp-targets=spir64 …
Windows*:  > ifx /Qopenmp-target-do-concurrent /Qiopenmp /Qopenmp-targets:spir64 …
```

OpenMP* におけるデバイスオフロード

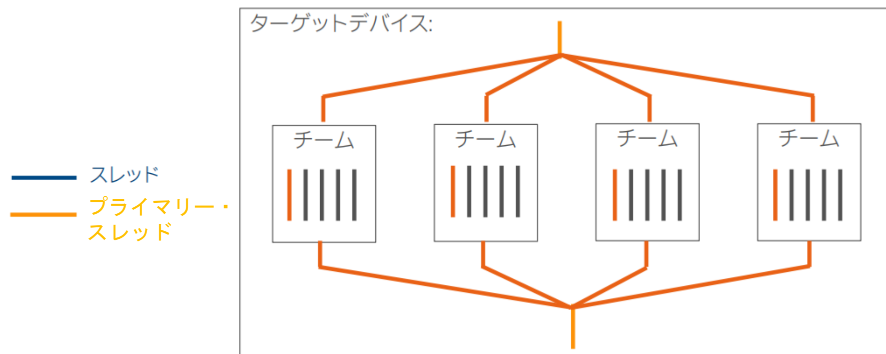
```
!$omp target map(tofrom:is_cpu)
is_cpu=omp_is_initial_device()
!$omp parallel do
do i=1,N
    x(i) = x(i) * 2
end do
!$omp end parallel do
!$omp end target
```

- target 構造
  - ホストからデバイスへの実行を指定
- map 節
  - ホスト/デバイス間でタイプによるデータ転送を手動で指定

主なマップタイプ	転送	説明
to	ホスト→デバイス	target 領域でデバイスにコピー
from	デバイス→ホスト	target 領域の完了時にホストへコピー
tofrom	ホスト⇔デバイス	target 領域でデバイスにコピー その後、完了時にホストへ戻す

GPU での実行を考慮する

- team 構造
  - プライマリー･スレッドとワーカースレッドで構成される 1 つ以上のチームを作成
- distribute 構造
  - ループ反復処理を各チームに分散



GPU での実行を考慮する

```
!$omp target teams distribute num_teams(NUM_BLOCKS) map(tofrom: y) map(to: x)
do ib=1,ARRAY_SIZE, NUM_BLOCKS
        do i=ib, ib+NUM_BLOCKS-1
                y(i) = a*x(i) + y(i)
        end do
end do
!$omp end target teams distribute
```

ARRAY_SIZE, a は定数

コンパイルオプション

- -fiopenmp (Linux*)、
  /Qiopenmp (Windows*)

  OpenMP* の認識を有効

  OpenMP* ディレクティブを基に並列コードの生成

- -fopenmp-targets=<T1> (Linux*)、
  /Qopenmp-targets=<T1> (Windows*)

  GPU ターゲットへのオフロードを有効

  T1: spir64 (JIT コンパイル)

```
ifx -fiopenmp -fopenmp-targets=spir64 main.f90
```

USM (統合共有メモリー) を使ったデータ制御

- allocate 句
  - ホストとデバイスで 1 つのアドレス空間を持ちます
  - データの転送を明記せず暗黙的な制御が可能
- パフォーマンスに影響しやすい
  - map 節による指定

```
!$omp allocate allocator(omp_target_shared_mem_alloc)
    allocate(x(N))
```

```
!$omp target
!$omp teams distribute parallel do
    do i=1,N
        x(i) = x(i) * 2
    end do
!$omp end target
```

ターゲットデバイスの選択

- device 節
  - 特定のデバイスでの実行を指定します
  - 複数の GPU を搭載した環境向け

```
!$omp target device(1)
```

- OMP_TARGET_OFFLOAD 環境変数
  - GPU へのオフロードを制御
  - OMP_TARGET_OFFLOAD={MANDATORY, DISABLED, DEFAULT}
    - デフォルト: DEFAULT

非同期で実行

- target 構造における実行が完了するまでホスト側は待機します
- nowait 節
  - target 構造の処理を非同期に実行します
  - taskwait 構造により同期します

```
!$omp task
call compute()
!$omp end task

!$omp target map(to:x1,y1) nowait
call compute_1(x1, y1, N)
!$omp end target

!$omp target map(to:x2,y2) nowait
call compute_2(x2, y2, N)
!$omp end target

!$omp taskwait
```

解析/デバッグツール
ハードウェアを最大限に活用

設計

インテル® Advisor

- コードを GPU へ効率良くオフロード
- CPU/GPU コードの計算とメモリーを最適化
- ベクトル並列処理をさらに有効にして効率を向上
- スレッド化されていないアプリケーションに効率的なスレッド化を追加

デバッグ

インテル® ディストリビューションの GDB

- CPU、GPU、FPGA を含むさまざまなアクセラレーターをサポート
- SYCL*、C、C++、OpenMP*、Fortran クロスアーキテクチャー・アプリケーションの詳細なシステム全体のデバッグが可能
- Microsoft* Visual Studio*、VS Code、Eclipse* との IDE 統合

チューニング

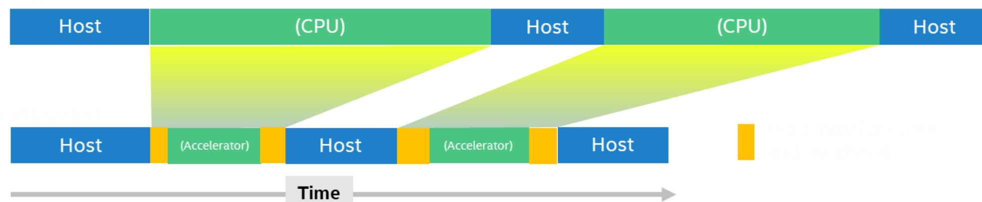
インテル® VTune™ プロファイラー

- GPU、CPU、FPGA 向けにチューニング
- オフロードのパフォーマンスを最適化
- SYCL*、C、C++、Fortran、Python*、Go*、Java*、混在言語をサポート

インテル® Advisor の GPU オフロード支援

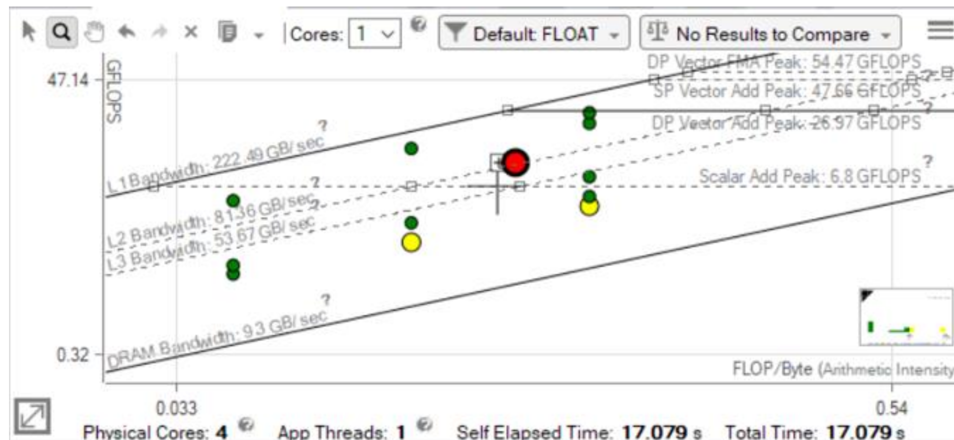
オフロード・アドバイザー

- GPU にオフロードした際のパフォーマンスを推定

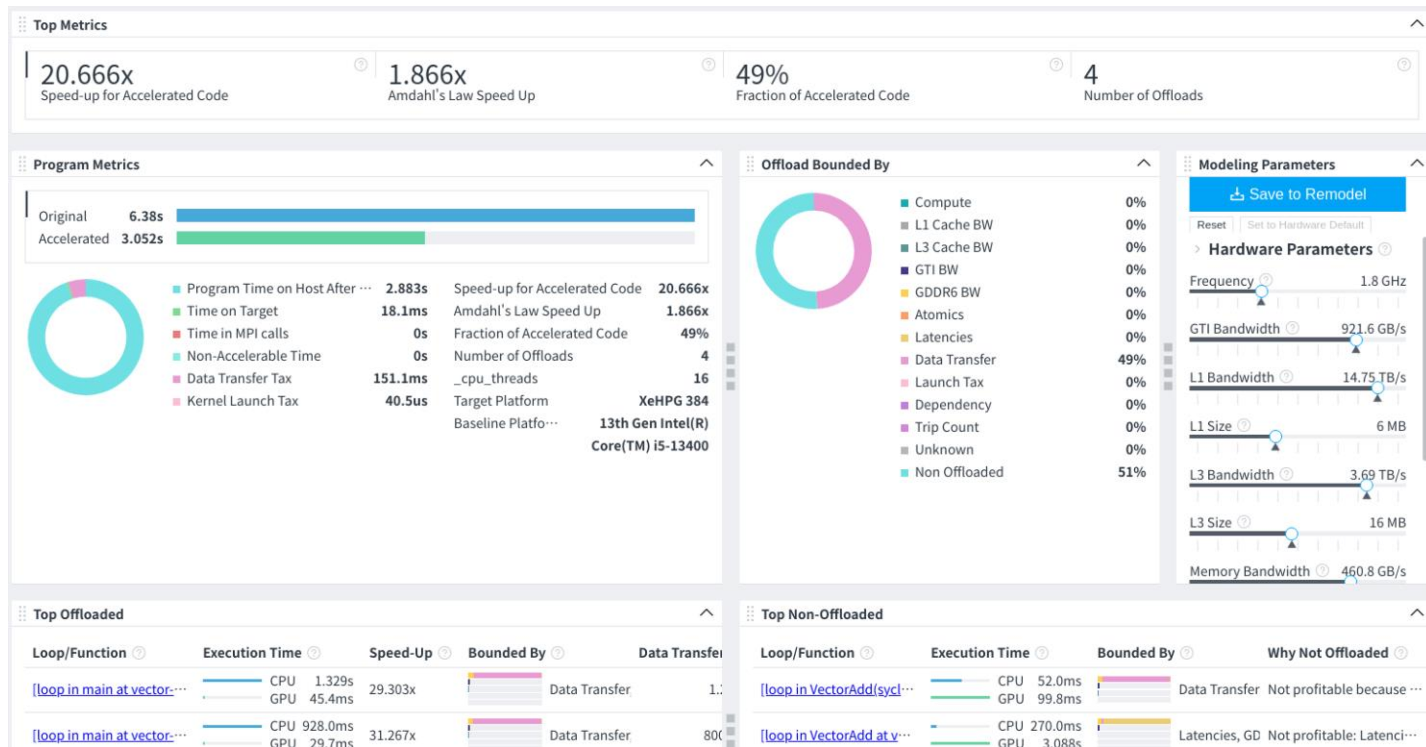


ルーフライン解析

- ターゲット環境における計算リソースの使用状況を可視化

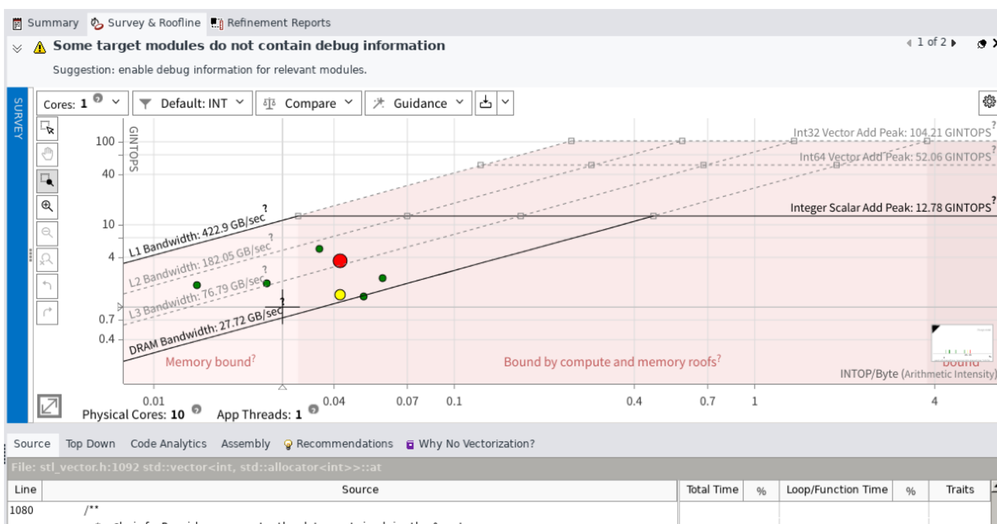


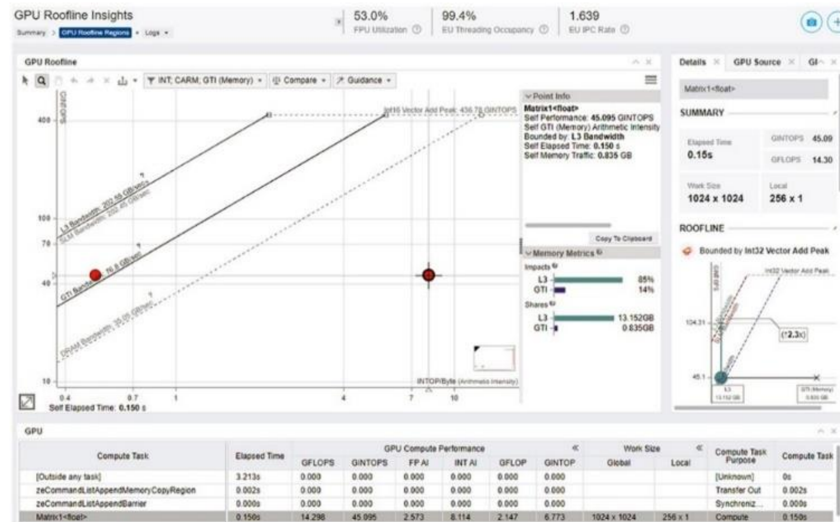
オフロード·アドバイザー



ルーフライン解析

- CPU と GPU のルーフラインを表示
  - ターゲット環境に対するパフォーマンスを可視化

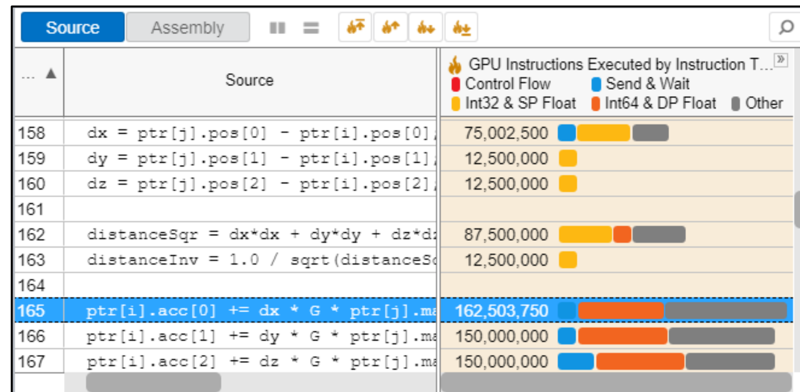


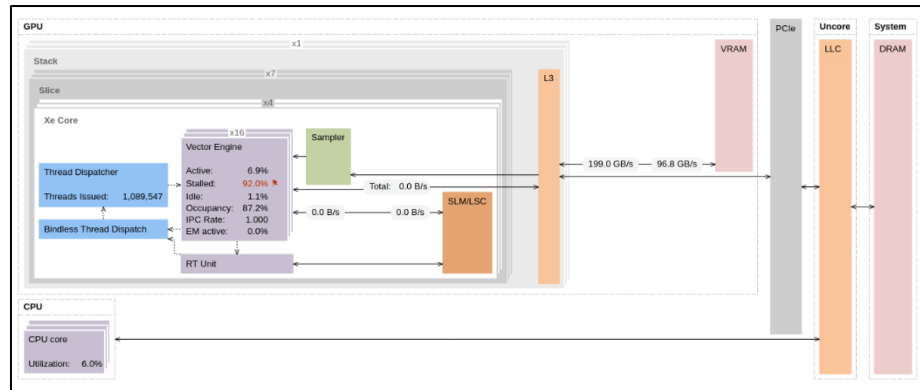


インテル® VTune™ プロファイラー

GPU アプリケーション向け解析タイプ

- GPU Offload
  - CPU 依存か GPU 依存か特定するための解析タイプ
- GPU Compute/Media Hotspots
  - 時間のかかる GPU の処理計算がアルゴリズムもしくはメモリーの利用に依存しているか確認





GPU Compute/Media Hotspots

■ 実行ユニットのストールを状況別に可視化

Grouping: GPU Adapter / GPU Stack / Computing Task / Function / Call Stack

GPU Adapter / GPU Stack / Computing Task / Function / Call Stack	Work Size		Stall Count by Stall Type								
	Global ▲	Local	Active	Other	Control	Pipe	Send	Dist or Acc	SBID	Synchronization	Instruction Fetch
▼ 0:58:0.0 : Display controller: Intel C			5.3%	0.2%	0.0%	1.4%	0.0%	0.5%	42.5%	0.0%	0.1%
▶ GPU Stack 0			2.7%	0.1%	0.0%	0.7%	0.0%	0.2%	21.1%	0.0%	0.0%
▶ GPU Stack 1			2.6%	0.1%	0.0%	0.7%	0.0%	0.2%	21.5%	0.0%	0.0%
▼ 0:154:0.0 : Display controller: Intel (			5.7%	0.2%	0.0%	1.3%	0.0%	0.5%	42.3%	0.0%	0.1%
▶ GPU Stack 0			2.9%	0.1%	0.0%	0.6%	0.0%	0.2%	21.2%	0.0%	0.0%
▶ GPU Stack 1			2.9%	0.1%	0.0%	0.6%	0.0%	0.2%	21.1%	0.0%	0.1%

GPU Compute/Media Hotspots (preview)

Analysis Configuration | Collection Log | Summary | Graphics

Elapsed Time: 11.299s

GPU Time: 2.426s

XVE Array Stalled/Idle: 94.5% of Elapsed time with GPU busy

Analyze the average value of the XVE Array Stalled/Idle metric. Understand why XVEs were waiting for resources and not computing. This metric is critical for compute-bound applications. Explore typical reasons for this inefficiency.

GPU L3 Bandwidth Bound: 2.6% of peak value

Identify whether performance of your code executing on the GPU is bounded by GPU L3 bandwidth.

Hottest GPU Computing Tasks Bound by GPU L3 Bandwidth

Occupancy: 70.5% of peak value

Identify computing tasks with low occupancy that were too large or small to render the XVE array idle, while waiting for the scheduler. Frequent SLM accesses and barriers can affect the maximum possible occupancy.

Hottest GPU Computing Tasks with Low Occupancy

This section lists the most active computing tasks running on the GPU with a low Occupancy, sorted by the Total Time.

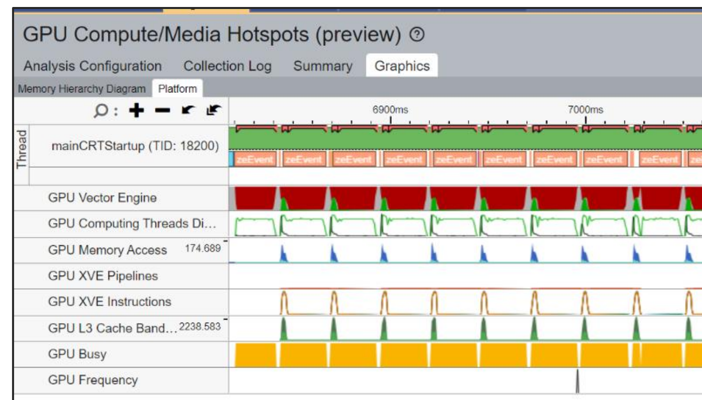
Computing Task	Total Time	SIMD Width	Peak XVE Threads Occupancy	Occupancy	SIMD Utilization
MandelParallel::Evaluate(sycl::_V1::queue&)::{lambda(sycl::_V1::handler&)#1}::operator()(sycl::_V1::handler&) const::{lambda()#1}	0.303s	8	100.0%	86.1%	100.0%

*N/A is applied to non-summable metrics

Bandwidth Utilization Histogram

Explore bandwidth utilization over time using the histogram and identify memory objects or functions with maximum contribution to the high bandwidth utilization.

Bandwidth Domain: GPU Memory Read Bandwidth, GB/sec



まとめ

- マルチアーキテクチャー・プログラミングのすすめ
  - C++、Fortran 言語で GPU の利用
- インテル® oneAPI ツールキットは DPC++ (C++ with SYCL*) および Fortran 言語のプログラムに対して GPU などのアクセラレーターを利用する手段を提供します
- 最適化を支援するインテル® Advisor および インテル® VTune™ プロファイラーを活用ください

参考資料

- Get Started with the Intel® oneAPI DPC++/C++ Compiler (英語)
- C/C++ or Fortran with OpenMP* Offload Programming Model (英語)

- Get Started with Intel® Advisor (英語)
- Intel® VTune™ Profiler - GPU Offload Analysis (英語)

日本語情報

- iSUS : 日本語版パッケージ & ドキュメント

お問い合わせはこちらまで

https://www.xlsoft.com/jp/qa

© 2023 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、一般に各社の商標または登録商標です。
製品および性能に関する情報: 性能は、使用状況、構成、その他の要因によって異なります。詳細については、http://www.intel.com/PerformanceIndex/ (英語) を参照してください。
Copyright © 2023 XLsoft Corporation. XLsoft のロゴ、XLsoft は XLsoft Corporation の商標です。