# インテル® ソフトウェア開発ツール: コンパイラーとツールのアップデート

エクセルソフト株式会社

2025/09/11

# はじめに

- ご紹介内容は 2025年8月時点の情報を基にしています
- この資料には 2025/1/23 に開催された
  「インテル® ソフトウェア開発ツール 2025 リリースセミナー」の情報と
  同じものを含みます

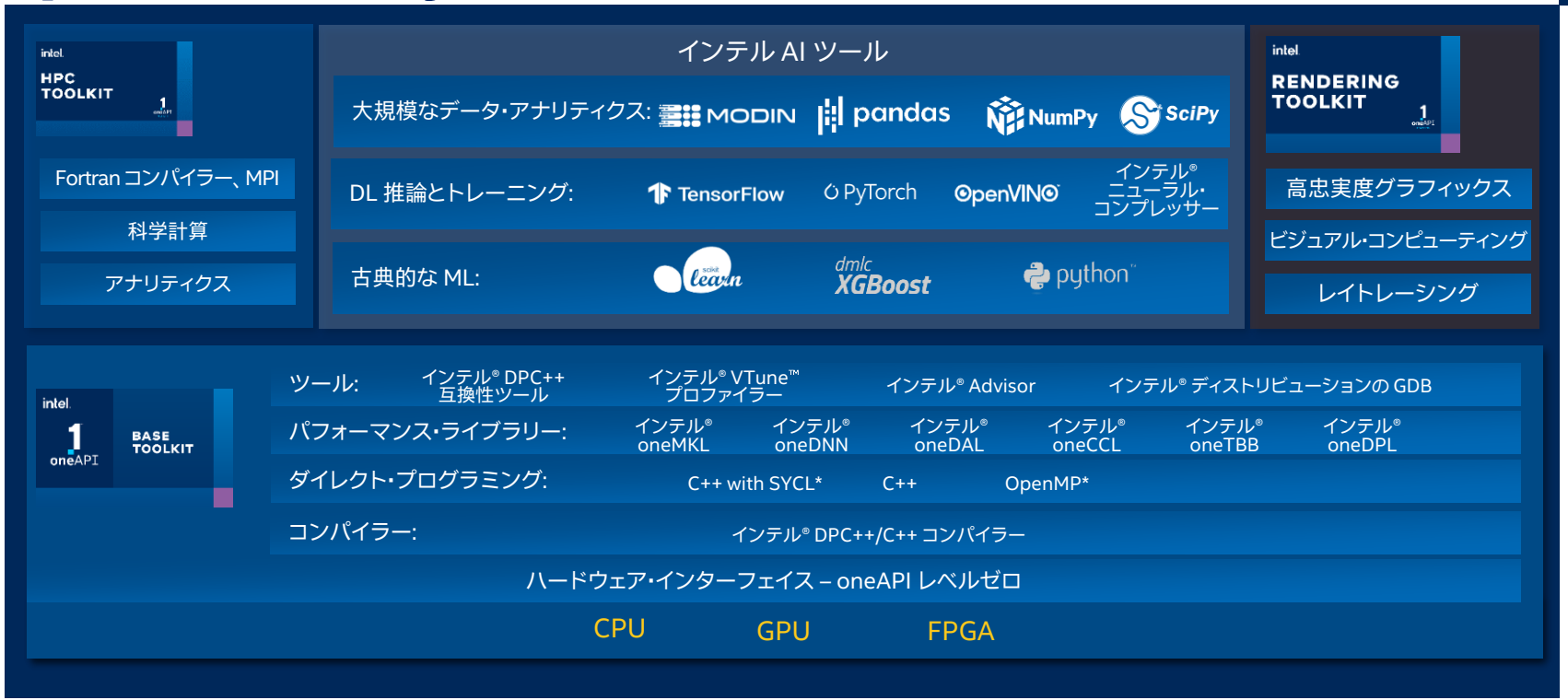
# バージョン 2025 について

- 2024年11月にバージョン 2025 がリリースされ、
  現在は 2025.2.0 が最新のバージョンです。
- 2025 リリースにともないバージョン 2024.x.x は
  旧バージョンとなりました
  - ✓ Intel Registration Center もしくは
    各種パッケージ・マネージャーからダウンロードできます
- メジャーリリースとなるため API やランタイムに旧バージョンとの互換性がない場合があります

# ご紹介内容

- インテル® ソフトウェア開発ツールの概要
- インテル® ソフトウェア開発ツール 2025 アップデート
  - ✓ コンパイラー
  - ✓ ツール、ライブラリー
- サポートについてのご案内

# インテル® ソフトウェア開発ツール powered by oneAPI



# インテル® oneAPI ベース & HPC ツールキット


### コンパイラーや実行環境

- インテル® Fortran コンパイラー
- インテル® oneAPI DPC++/C++ コンパイラー
- インテル® ディストリビューションの Python*

### ツール/ライブラリー

- インテル® MPI ライブラリー
- 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 DPC++ ライブラリー (インテル® oneDPL)
- 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 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 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® 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® DPC++ 互換性ツール

### 解析/デバッグツール

- インテル® VTune™ プロファイラー
- インテル® Advisor
- インテル® ディストリビューションの GDB

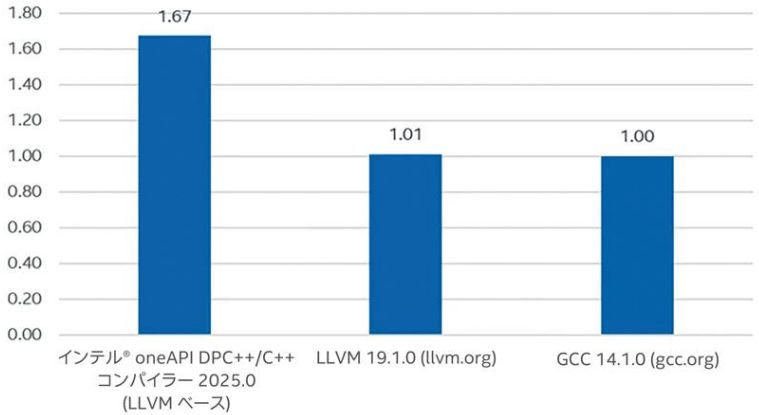
■ インテル® **HPC** ツールキットにのみ含まれるもの

個別ダウンロードが必要なコンポーネントを含みます

# インテル® コンパイラーが Linux* 上の C++ アプリケーションのパフォーマンスを向上

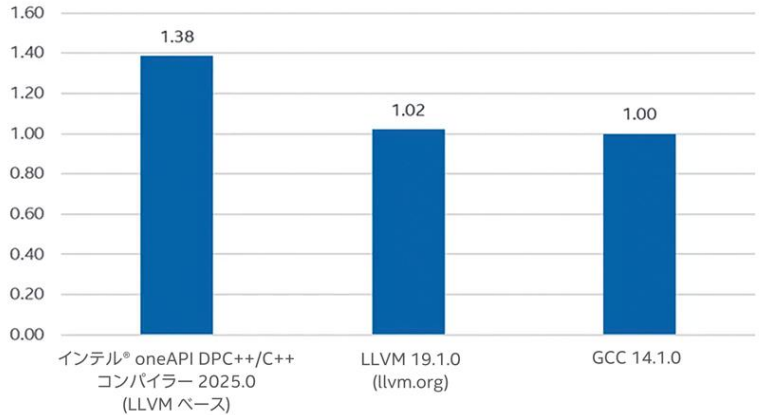
## インテル® Xeon® 6980P プロセッサー上でほかのコンパイラーと比較したパフォーマンスの優位性

浮動小数点レートのパフォーマンスの比較 (推定値)
GCC 14.1 = 1.00、値が大きいほうが良い




推定値: SPECrate* 2017 Floating Point Suite (baseline) の C/C++ ワークロードの幾何平均の内部測定

整数レートのパフォーマンスの比較 (推定値)
GCC 14.1 = 1.00、値が大きいほうが良い




推定値: SPECrate* 2017 Integer Suite (baseline) の C/C++ ワークロードの幾何平均の内部測定

**テスト実施日:** 性能の測定結果は **2024年10月20日時点のインテルの社内テスト**に基づいています。また、現在公開中のすべてのセキュリティー・アップデートが適用されているとは限りません。

**システム構成とワークロード設定:** インテル® Xeon® 6980P CPU、B2 ステッピング、2.0GHz、2 ソケット、インテル® ハイパースレッディング・テクノロジー有効、インテル® ターボ・ブースト・テクノロジー有効、64G x24 DDR5 6400 (1DPC)、ソフトウェア: インテル® 64 対応アプリケーション用インテル® oneAPI DPC++/C++ コンパイラー v2025.0.0 ビルド 20240928、GCC 14.1.0、LLVM 19.1.0。Ubuntu* 24.04 LTS、6.8.0-45-generic。SPECint_rate_base_2017 コンパイラー・オプション: インテル® oneAPI DPC++/C++ コンパイラー: -xsapphirerapids -O3 -ffast-math -flto -mfpmath=sse -funroll-loops -qopt-mem-layout-trans=4。GCC: -march=sapphirerapids -mfpmath=sse -Ofast -funroll-loops -flto。LLVM: -march=sapphirerapids -mfpmath=sse -Ofast -funroll-loops -flto。jemalloc 5.0.1 をインテル® コンパイラー、GCC、LLVM に使用。SPECfp_rate_base_2017 コンパイラー・オプション: インテル® oneAPI DPC++/C++ コンパイラー: -xsapphirerapids -mprefer-vector-width=512 -Ofast -ffast-math -flto -mfpmath=sse -funroll-loops -qopt-mem-layout-trans=4。GCC: -march=sapphirerapids -mfpmath=sse -Ofast -funroll-loops -flto。LLVM: -march=sapphirerapids -mfpmath=sse -Ofast -funroll-loops -flto。jemalloc 5.0.1 をインテル® コンパイラー、GCC、LLVM に使用。

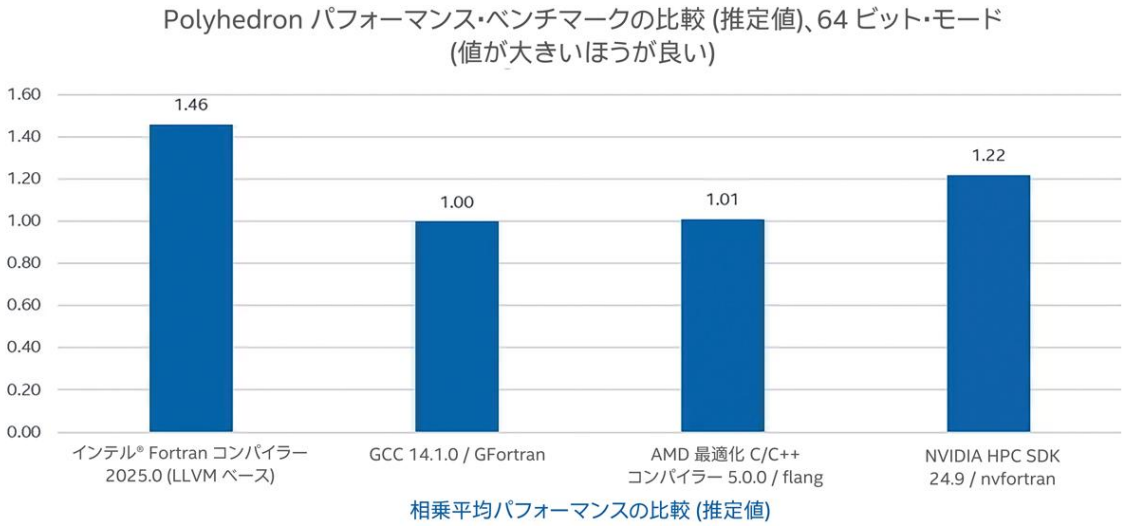
性能の測定結果はシステム構成の日付時点のテストに基づいています。また、現在公開中のすべてのアップデートが適用されているとは限りません。詳細については、公開されている構成情報を参照してください。絶対的なセキュリティーを提供できる製品またはコンポーネントはありません。

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、www.intel.com/PerformanceIndex を参照してください。実際の費用と結果は異なる場合があります。

(参照) インテル® oneAPI DPC++/C++ コンパイラー ベンチマーク

# インテル® Fortran コンパイラーが Linux* 上のアプリケーションのパフォーマンスを向上

## インテル® Xeon® 6980P プロセッサー上での Polyhedron Fortran ベンチマークにおけるパフォーマンスの優位性




**テスト実施日:** 性能の測定結果は **2024年10月20日時点のインテルの社内テスト**に基づいています。また、現在公開中のすべてのセキュリティー・アップデートが適用されているとは限りません。

**システム構成とワークロード設定:** インテル® Core™ Ultra 7 268V プロセッサー、2.20GHz、32G DDR5。ソフトウェア: インテル® 64 対応アプリケーション用インテル® Fortran コンパイラー v2025.0.0 ビルド 20240928、GCC 14.1.0 / GFortran、AMD 最適化 C/C++ コンパイラー 5.0.0 / flang、AMD Clang バージョン 17.0.6 (Clang: AOCC_5.0.0-Build#1377 2024_09_24)、NVIDIA HPC SDK 24.9 / nvfortran、Ubuntu* 22.04 LTS 6.11.0-rc4+prerelease1785+。コンパイラー・オプション: インテル® Fortran コンパイラー: -Ofast -xCORE-AVX512 -flto -nostandard-realloc-lhs。GCC / Gfortran: -Ofast -mfpmath=sse -flto -march=skylake-avx512 -funroll-loops。AMD 最適化 C/C++ コンパイラー / flang: -O3 -ffast-math -march=znver5 -fveclib=AMDLIBM -flto -mllvm -unroll-aggressive -mllvm -unroll-threshold=500。NVIDIA HPC SDK / nvfortran: -fast -Mipa=fast,inline -Mallocatable=03 -Mfprelaxed -Mstack_arrays。

性能の測定結果はシステム構成の日付時点のテストに基づいています。また、現在公開中のすべてのアップデートが適用されているとは限りません。詳細については、公開されている構成情報を参照してください。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。

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、www.intel.com/PerformanceIndex を参照してください。実際の費用と結果は異なる場合があります。

# インテル® ソフトウェア開発ツール 2025 アップデート 全体

- 最新のインテルのハードウェアに対応
  - ✓ インテル® Core™ Ultra プロセッサー (開発コード名: Lunar Lake)
  - ✓ インテル® Xeon® 6 プロセッサー (開発コード名: Granite Rapids)
    - 例: コンパイラー・オプションでハードウェア指定が可能
    - Linux コマンド: -x***lunarlake***、-march=***graniterapids***
- 一部のコンポーネントの廃止
- ツールのダウンロード形態の追加
- 新しい OS、環境に関する動作要件のサポートは下記
  - ✓ インテル® oneAPI ベース・ツールキット 2025.2 動作環境
  - ✓ インテル® HPC ツールキット 2025.2 動作環境

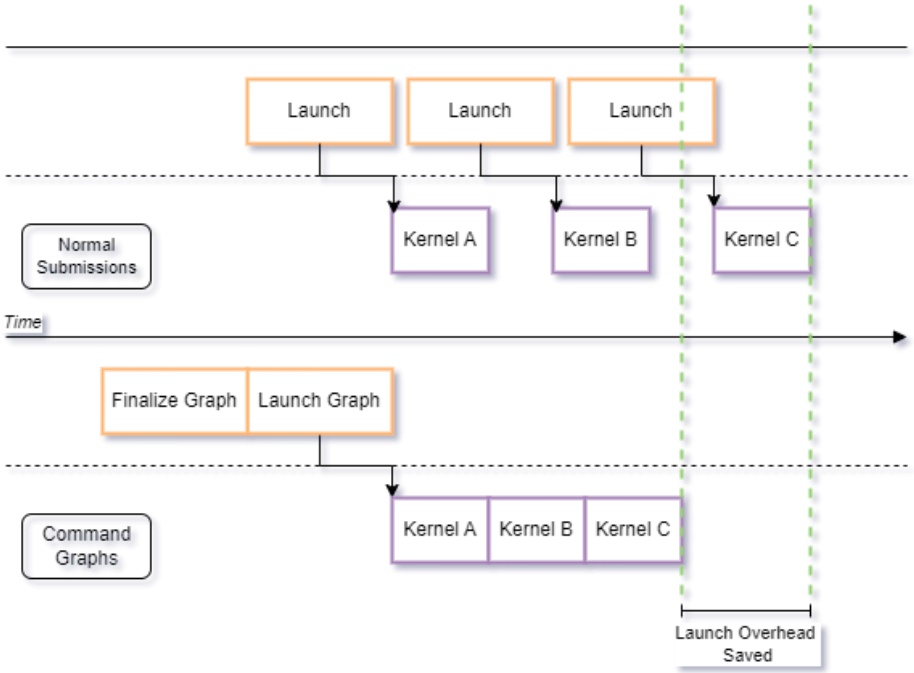

P-cores 搭載インテル® Xeon® 6 プロセッサー

# インテル® ソフトウェア開発ツール 2025
# DPC++/C++ コンパイラー

- SYCL Graph 機能のアップデート
  - ✓ ホスト-タスクノードの更新機能
  - ✓ 動的コマンドグループ
- LLVM サニタイザー機能にデバイスコードの対応を追加
  - ✓ インテル® oneAPI DPC++/C++ コンパイラーでサニタイザーを使用してバグを素早く検出
- SYCL とグラフィックス API の連携による画像処理最適化
- ハードウェア プロファイルに基づく最適化の改善
- SYCL カーネルを含む FAT バイナリーのサイズを圧縮する機能を追加
  - ✓ -offload-compress オプション

# SYCL Graph 機能

- SYCL プログラムの中で複数の処理(カーネルやメモリー操作)をグラフ構造で管理
  - ✓ 計算カーネルとメモリー操作の依存関係の計算グラフを定義し、バッチ化
- 各処理を「ノード」として登録、依存関係を明示的に記述することで、最適な順序で実行可能
- カーネル起動のオーバーヘッドを削減し、高頻度で同じ処理を繰り返す場合の実行効率を高める
- SYCL Graph Usage Guide and Examples (llvm) (英語)



# SYCL Graph サンプル: 記録と再生 API

グラフの宣言

グラフの記録開始

ノード A
(キューに送信された
コマンド)

ノード B

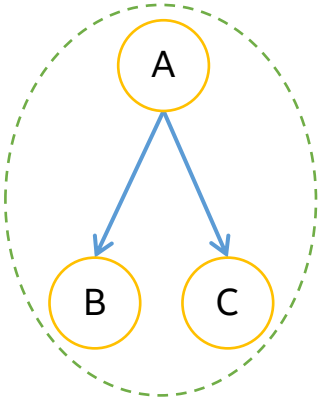
グラフの記録終了

ファイナライズ

グラフをキューに送信

```
int main() {
    namespace sycl_ext = sycl::ext::oneapi::experimental;

    sycl::queue Queue;
    sycl_ext::command_graph Graph(Queue.get_context(), Queue.get_device());

    Graph.begin_recording(Queue);

    sycl::event HostWorkEvent = Queue.submit([&](sycl::handler& CGH) {
        CGH.host_task([=]() {
            do_some_host_work();
            });
        });

    Queue.submit([&](sycl::handler& CGH) {
        CGH.parallel_for(...);
          // ノード B の処理
        });

    Graph.end_recording(Queue);

    auto ExecGraph = Graph.finalize();

    Queue.ext_oneapi_graph(ExecGraph).wait();

            // …
}
```

# SYCL Graph サンプル: 明示的指定




グラフの宣言

ノード A
(グラフに追加)

ノード B
(ノード A に依存)

ノード C
(ノード A に依存)

ファイナライズ
グラフをキューに送信

```
int main() {
    namespace sycl_ext = sycl::ext::oneapi::experimental;

    sycl::queue Queue;
    sycl_ext::command_graph Graph(Queue.get_context(), Queue.get_device());

    auto node_a = Graph.add([&](sycl::handler& h) {
        h.parallel_for(…) {
            //ノードAの処理
        });
     });

    auto node_b = g.add([&](sycl::handler& h) {
        h.parallel_for(…) {
            //ノードBの処理
          });
        }, { sycl_ext::property::node::depends_on(node_a) });

    auto node_c = g.add([&](sycl::handler& h) {
        h.parallel_for(…) {
            //ノードCの処理
            });
        }, { sycl_ext::property::node::depends_on(node_a) });

    auto ExecGraph = Graph.finalize();
    Queue.ext_oneapi_graph(ExecGraph).wait();
    // …
}
```

# インテル® コンパイラーのサニタイザー機能

- USM、SYCL バッファー、ローカルメモリーなどの領域で、
  未初期化メモリーやデータ競合、use-after-free などの問題を検出可能
  - ✓ GPU や OpenMP デバイスコードに新しく対応
- コンパイラー・オプション: -fsanitize=<sanitizer>
  - ✓ -fsanitize=address
    - バッファー・オーバーフロー/アンダーフロー
    - メモリーリーク (Linux のみ)
  - ✓ -fsanitize=memory (Linux のみ)
    - 初期化されていない変数
  - ✓ -fsanitize=thread (Linux のみ)
    - スレッド間のデータ競合

# 例: アドレスサニタイザー

```c
#include <stdlib.h>
int main() {
    char* x = (char*)malloc(10 * sizeof(char*));
    free(x);
    return x[5];
}
```

コード: 出典 (英語)

```
xlsoftkk@xlsta750:~/Desktop/sanitizer$ icx -fsanitize=address use-after-free.c
xlsoftkk@xlsta750:~/Desktop/sanitizer$ ./a.out
=================================================================
==134663==ERROR: AddressSanitizer: heap-use-after-free on address 0x6ec0d19e0025 at pc 0x00000051abac bp 0x7ffc30dcf260
sp 0x7ffc30dcf258
READ of size 1 at 0x6ec0d19e0025 thread T0
   #0 0x00000051abab in main (/home/xlsoftkk/Desktop/sanitizer/a.out+0x51abab)
   #1 0x7250d262a1c9 in __libc_start_call_main csu/../sysdeps/nptl/libc_start_call_main.h:58:16
   #2 0x7250d262a28a in __libc_start_main csu/../csu/libc-start.c:360:3
   #3 0x00000042b3f4 in _start (/home/xlsoftkk/Desktop/sanitizer/a.out+0x42b3f4)

0x6ec0d19e0025 is located 5 bytes inside of 80-byte region [0x6ec0d19e0020,0x6ec0d19e0070)
freed by thread T0 here:
   #0 0x0000004d6fa2 in free /netbatch/donb81172_00/dir/workspace/NIT/xmain-
rel/LX/xmainefi2linux_release/ws/icsws/llvm/compiler-rt/lib/asan/asan_malloc_linux.cpp:51:3
   #1 0x00000051ab7b in main (/home/xlsoftkk/Desktop/sanitizer/a.out+0x51ab7b)
   #2 0x7250d262a28a in __libc_start_main csu/../csu/libc-start.c:360:3
   #3 0x00000042b3f4 in _start (/home/xlsoftkk/Desktop/sanitizer/a.out+0x42b3f4)

previously allocated by thread T0 here:
   #0 0x0000004d7248 in malloc /netbatch/donb81172_00/dir/workspace/NIT/xmain-
rel/LX/xmainefi2linux_release/ws/icsws/llvm/compiler-rt/lib/asan/asan_malloc_linux.cpp:67:3
   #1 0x00000051ab70 in main (/home/xlsoftkk/Desktop/sanitizer/a.out+0x51ab70)
```

# デバイス側のサニタイザー

- 前述の通り、オフロードコードについても新しく対応
  - ✓ Memory、Thread の GPU 対応はインテル® データセンター GPU のみ
- -Xarch_device で、明示的にデバイスコードを対象とするよう指示
  - ✓ icpx -fsycl -Xarch_device -fsanitize=address sample.cpp

```
xlsoftkk@xlsta750:~/Desktop/sanitizer$ icpx -fsycl -Xarch_device -
fsanitize=address  asan_sample.cpp
xlsoftkk@xlsta750:~/Desktop/sanitizer$ ./a.out
==== DeviceSanitizer: ASAN

====ERROR: DeviceSanitizer: out-of-bounds-access on Memory Buffer
(0xffffd556aa0a00c0)
WRITE of size 4 at kernel <typeinfo name for
main::'lambda'(sycl::_V1::handler&)::operator()(sycl::_V1::handler&)
const::'lambda'(sycl::_V1::id<1>)> LID(0, 0, 0) GID(0, 0, 0)
  #0 typeinfo name for
main::'lambda'(sycl::_V1::handler&)::operator()(sycl::_V1::handler&)
const::'lambda'(sycl::_V1::id<1>) <unknown file>:0
```

```
const size_t N = 100;

sycl::queue myQueue;
int* data = sycl::malloc_shared<int>(N, myQueue);

myQueue.parallel_for(N, [=](sycl::id<1> idx) {
  data[idx] = idx;
  if (idx == N - 1) {
    data[idx + 1] = 999;
  }
}).wait();
```

# ハードウェア・プロファイルに基づく最適化 (HWPGO)

- クラシック・コンパイラーにおける PGO 機能の強化版
  - ✓ 特に分岐予測の最適化について有益とされています
- インテルプロセッサーのパフォーマンス監視ユニット (PMU) を利用
  - ✓ さまざまなイベントの情報を収集
    - リタイアされた命令、キャッシュミス、分岐予測ミス、など…
- 分岐予測に関する HWPGO の手順
  - ✓ -fprofile-sample-generate でコンパイル
  - ✓ サンプリングデータを収集
  - ✓ llvm-profgen を使用してサンプリングデータからプロファイルを生成
  - ✓ -fprofile-sample-use でプロファイル情報を基にリコンパイル
- インテルのハードウェア・ベースのプロファイルに基づく最適化 (PGO)

# SYCL とグラフィックス API の連携による画像処理最適化

- インテル® Arc™ GPU 向けバインドレス・テクスチャーを実装
  - ✓ GPU グラフィックス・プログラミングにおいて、テクスチャーを利用する際の「バインド」操作を不要に
- Vulkan および DirectX グラフィックス API との SYCL 相互運用性が強化
  - ✓ CPU と GPU 間の不要なメモリーコピーの削減

# インテル® ソフトウェア開発ツール 2025
# Fortran コンパイラー

- Fortran 2023 標準規格の対応を拡張
  - ✓ 新しい純粋組み込みサブルーチンの追加や挙動変更
    例: SPLIT や TOKENIZE の追加、IEEE_ARITHMETIC 機能の挙動変更
  - ✓ Conformance, Compatibility, and Fortran Features (Intel.com) (英語)
- OpenMP* 6.0 / 5.1 のサポートを拡張
- -check all の動作は -check nouninit を含むように変更
- -fopenmp-offload-mandatory は OpenMP* TARGET 領域についてデバイスコードのみ生成するように指示可能
  - ✓ -fopenmp-targets オプションによる OpenMP* オフロードが必須

# インテル® ソフトウェア開発ツール 2025
# ライブラリー

- インテル® oneAPI マス・カーネル・ライブラリー
  - ✓ SYCL ベースの分散 DFT(FFT)API が追加され、複数 GPU で FFT の計算が可能に
  - ✓ GEMM(行列積)API が、8 ビット浮動小数点形式 (e4m3/e5m2) に対応
  - ✓ インテル® AMX もしくは インテル® AVX-512 を利用した BLAS 関数の性能改善

- インテル® MPI ライブラリー

  最新バージョンは 2021.16.1
  - ✓ MPI 4.1 完全対応
  - ✓ インテル® Xeon® 6 向けチューニング

    スケールアウト・スケールアップ両対応のチューニング

    非対称な CPU コア数における CPU Pinning の改善

# インテル® ソフトウェア開発ツール 2025 ツール

- インテル® VTune™ プロファイラー
  - ✓ 最新のハードウェアや OS への対応
    - インテル® Core™ Ultra プロセッサー 200V シリーズ (開発コード名: Lunar Lake)
    - インテル® Core™ Ultra プロセッサー 200S シリーズ (開発コード名: Arrow Lake-S)
    - インテル® Xeon® 6 プロセッサー (開発コード名: Granite Rapids)
    - インテル® Xeon® 6 SoC (開発コード名: Granite Rapids-D)
    - Rocky Linux 9
  - ✓ Python 3.12 のプロファイルをサポート
  - ✓ サポート終了について
    - Eclipse 統合は非推奨となり、次のリリースで削除予定
    - Ice Lake 以前の Xeon、10世代未満の Core プロセッサー

# 削除または非推奨となるコンポーネントのご案内

- バージョン 2025 より下記のコンポーネントがパッケージから削除されました
  - ✓ インテル® Fortran コンパイラー・クラシック
  - ✓ インテル® Inspector
    代替機能としてインテル® コンパイラーのサニタイザー機能もしくは外部ツールの Valgrind が提案されています
  - ✓ インテル® Trace Analyzer & Collector
    同等の機能を持つとされるサードパーティー製のツールの利用もご検討ください
    インテル® Trace Analyzer & Collector 製品終息 (EOL) のご案内
  - ✓ インテル® oneAPI DPC++/C++ コンパイラー用 FPGA サポートパッケージ
    今後は Altera が提供するツールによってサポートされます

# インテル® Fortran コンパイラー・クラシック (ifort) の廃止

- インテル® Fortran コンパイラー・クラシック (ifort) は 2025.0 で削除
  - ✓ ifort へのサポートは、削除とともに終息
- インテル® Fortran コンパイラー (ifx) への移行を推奨
  - ✓ インテル® Fortran コンパイラー・クラシックのフロントエンドと LLVM バックエンドを採用
  - ✓ 32ビット・アプリケーションのサポートを含みません
- ほとんどのオプションはそのまま受け付ける
  - ✓ 一方で、同じオプションでも動作/実装の違いがある
  - ✓ インテル® oneAPI ポーティング・ガイド (ifx) 日本語版

# パッケージに含まれるコンポーネントの変更

- バージョン 2025 の HPC ツールキットはベース・ツールキットのコンポーネントを同梱するようになりました
- ベース・ツールキット + HPC ツールキットの個別インストールは不要
  - ✓ HPC ツールキットのみで全コンポーネントをインストールできます
- 新しく 3 つのダウンロード・パッケージを提供するようになりました
  - ✓ コンパイラー + ライブラリーの小規模なパッケージです
    - インテル® C++ エッセンシャルズ
    - インテル® Fortran エッセンシャルズ
    - インテル® ディープラーニング・エッセンシャルズ

# 新しいダウンロード・パッケージ

## インテル® C++ エッセンシャルズ

- インテル® oneAPI DPC++/C++ コンパイラー
- インテル® oneAPI DPC++ ライブラリー
- インテル® DPC++ 互換性ツール
- インテル® oneTBB
- インテル® oneMKL

## インテル® Fortran エッセンシャルズ

- インテル® Fortran コンパイラー
- インテル® ディストリビューションの GDB
- インテル® oneMKL
- インテル® MPI ライブラリー

## インテル® ディープラーニング・エッセンシャルズ

- インテル® oneAPI DPC++/C++ コンパイラー
- インテル® oneAPI DPC++ ライブラリー
- インテル® oneDNN
- インテル® oneCCL
- インテル® oneMKL

# 宣伝: 技術サポートについて

- 弊社でお取り扱いしているインテル® oneAPI ツールキット製品は
  サポートサービスを提供する有償サポート製品です
- サポートサービスには旧バージョンのダウンロードおよび
  技術サポートを含みます
  - ✓ 技術サポートは製品に関するご質問についてお受けしています

    各コンポーネントの利用にあたって発生した問題に関する対応や
    動作要件の確認、最適化に関するご相談など
- 弊社が作成した資料や、過去に開催したウェビナーやイベントの録画なども
  ご覧いただけます

# お問い合わせはこちらまで

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