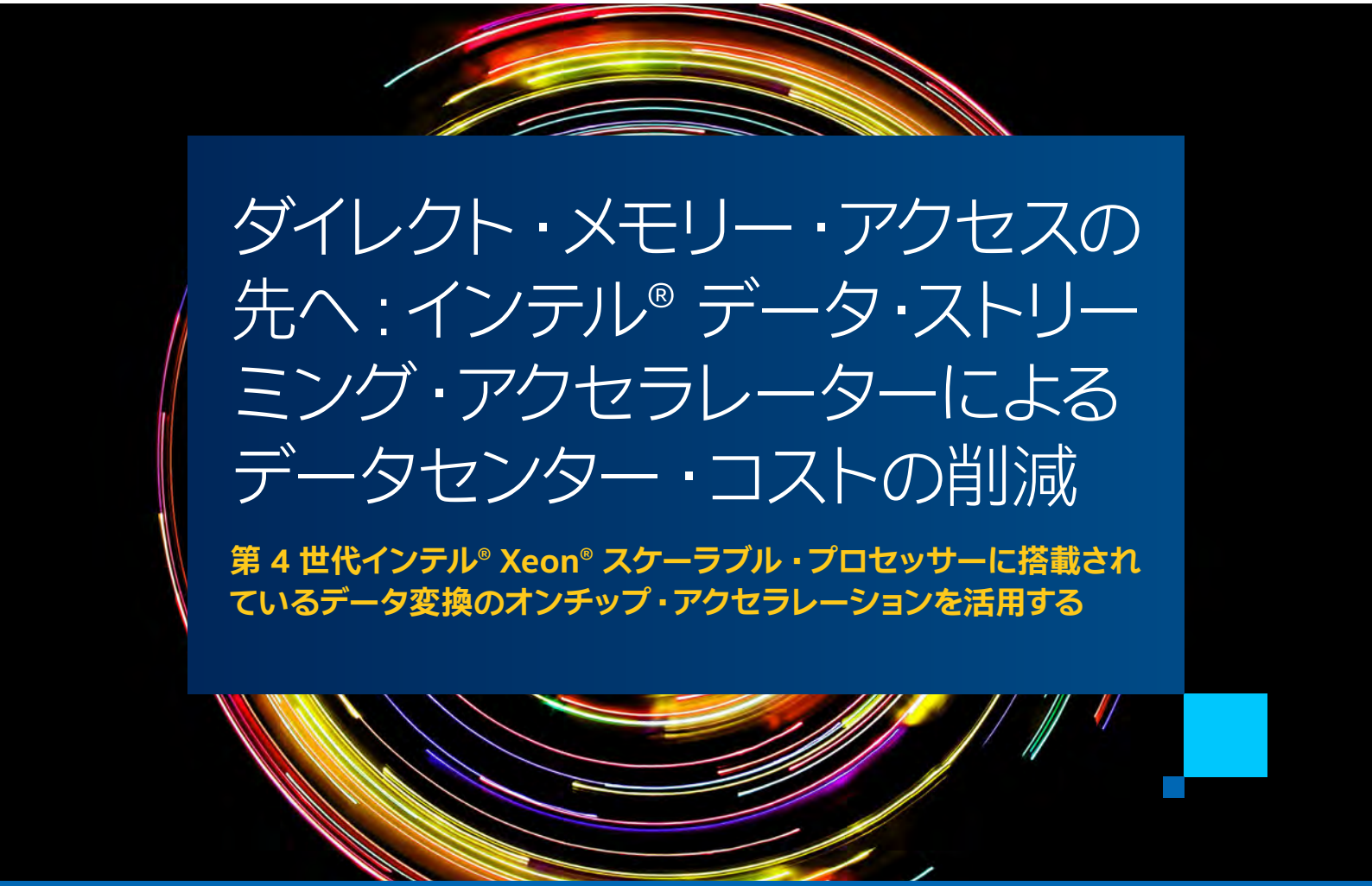

ダイレクト・メモリー・アクセスの先へ：インテル® データ・ストリーミング・アクセラレーターによるデータセンター・コストの削減

第 4 世代インテル® Xeon® スケーラブル・プロセッサーに搭載されているデータ変換のオンチップ・アクセラレーションを活用する

Sanjay Kumar インテル コーポレーション 主席エンジニア
Reese Kuper 同 研究インターン
Atul Kwatra 同 フェロー
Shibani Nataraja 同 製品マネージャー
Narayan Ranganathan 同 主席エンジニア
Rajesh Sankaran 同 シニアフェロー
Nikhil Rao、Ren Wang、および Yifan Yuan 同 科学研究員

最新のデータセンターでは、`memcpy()`、`memmove()`、ハッシュ、圧縮などの負荷の高いメモリー操作と変換を含む多くのアプリケーションが利用されるようになっています。アプリケーションには、データベース、画像処理、ビデオ転送、グラフ処理などが含まれますが、これらに限定されるものではありません。このようなメモリー操作と変換はデータセンターのインフラストラクチャー・ソフトウェアでも一般的であり、大量の CPU サイクルを消費します。これらのサイクルはアプリケーションの実行に使用される可能性があるため、「データセンター・タックス」（英語）と呼ばれることもあります。理想的には、このような操作は最適化されたハードウェア・エンジンにオフロードするべきです。

インテル® データ・ストリーミング・アクセラレーター

インテル® データ・ストリーミング・アクセラレーター（インテル® DSA）は、最新の第 4 世代インテル® Xeon® スケーラブル・プロセッサーに統合された、ハイパフォーマンスなデータコピーと変換のアクセラレーターであり、処理効率と実用性だけでなく、汎用性も提供します。インテル® DSA には、1 人以上のユーザーから送信されたワーク記述子を効率良く処理するハードウェア・コンポーネントが搭載されています。共有仮想メモリー（SVM）のサポートにより、これらのワーク記述子はユーザー・アプリケーションからメモリーマップ I/O（MMIO）レジスター経由でインテル® DSA に直接送信でき、ターゲットメモリー領域をオペレーティング・システムでピニングする必要はありません。ユーザーは、専用ドライバーで提供されるユーザー・インターフェイスを利用して、ニーズに基づいて計算リソースを直接構成することもできます。インテル® DSA は、新しくサポートされた操作により、さまざまな優れた機能を提供します（**表 1**）。多くのユーザー / カーネルのプロセスで、インテル® DSA が提供する機能を活用できることが期待されます。

タイプ	操作	説明
移動	メモリーコピー	ソースアドレスからディスネーション・アドレスへデータをコピー
	デュアルキャスト	ソースアドレスから 2 つの別のディスネーション・アドレスへデータを転送
	CRC 生成	転送されたソースデータの CRC チェックサムを作成
	データ整合性フィールド (DIF) 操作	ソースアドレスからディスネーション・アドレスへ転送されたソースデータの各 512/520/4096/4104 バイト・ブロックの DIF を確認、挿入、ストリップ、または更新
フィル	メモリーフィル	8/16 バイトの固定パターンで指定されたメモリー領域を連続してフィル
比較	メモリー比較	2 つのソースバッファーを相互に比較して同一かどうかをリターン
	パターン比較	ソースバッファーを 8 バイトのパターンと比較して領域全体がパターンと一致するかどうかをリターン
	デルタレコードの作成	2 つのソースバッファー (元のバッファーと変更されたバッファー) 間の差を含むデルタレコードを作成
	デルタレコードの適用	デルタレコードを元のバッファーにマージして変更されたバッファーのコピーをディスネーション・アドレスに作成
フラッシュ	キャッシュフラッシュ	キャッシュ階層から指定されたアドレス範囲内のすべてのキャッシュラインを退避

表 1. インテル® DSA でサポートされているデータ・ストリーミング操作

インテル® Data Accelerator Driver（IDXD）は、デバイスの初期化と管理を行うカーネル・モード・ドライバーであり、インテル® DSA の検出、初期化、構成の機能を提供します。アプリケーションは、これらの制御に、ユーザー空間 `libaccel-config` API ライブラリーを利用できます。データパスについては、アプリケーションのインテル® DSA インスタンスに低レイテンシー・アクセスを提供するため、IDXD は `mmap()` を介して MMIO ポータルを char デバイスとして利用できるようにします。これらの新機能と改良により、さまざまな分野（**表 2**）、特に中規模から大規模のデータサイズの操作（**図 1**）でインテル® DSA のパフォーマンスが同等のソフトウェアよりも大幅に向上していることが分かります。

分野	アプリケーションと操作の例
ネットワーク・スタックの高速化	DPDK、ソフトウェア仮想スイッチ、ビデオ転送
ストレージスタックの高速化	SPDK、NVMe-oF、DAOS
データセンター・コストの削減	VM の起動 / 移行、メモリー・コンパクション
HPC/ML の高速化	libfabric/MPI でのメモリー移動 / ゼロ化
ヘテロジニアス・メモリー管理の高速化	CXL/Pmem ベースの階層型メモリーシステムのページ移行

表 2. さまざまな分野におけるインテル® DSA の活用例

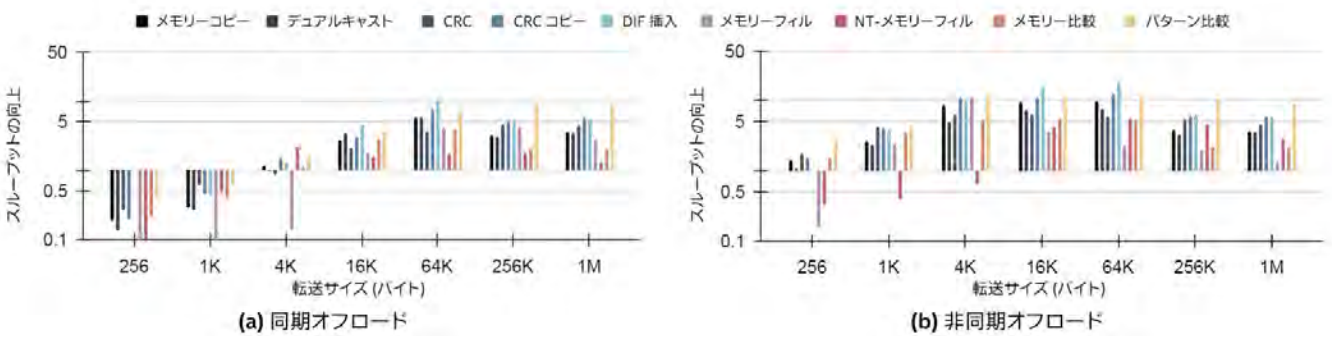

図 1. さまざまな転送サイズ（バッチサイズ : 1）における、対応するソフトウェアと比較したデータ・ストリーミング操作のスループットの向上。メモリーフィルと NT- メモリーフィルは、割り当て書き込みと非割り当て書き込みを指します（通常の `store` と `nt-store` に似ています）。

インテル® DSA の操作

ほかのアクセラレーターと同様に、インテル® DSA は記述子とワークキュー（WQ）を使用して CPU ソフトウェアと対話します（**図 2**）。

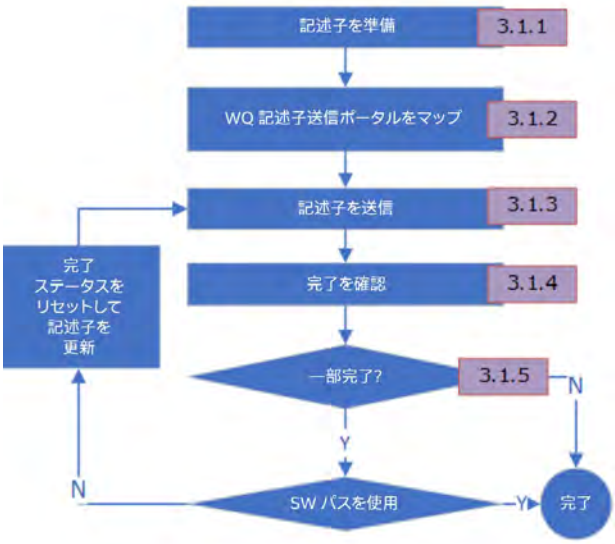

図 2. 記述子の処理手順

データの移動はインテル® DSA の最も一般的なユースケースです。データコピーを例に使用法を示します。キーと値ペアのサイズが大きなキー値格納アプリケーションでは、特定のキーの値を取得または更新するたびにメモリーコピーが発生し、大量の CPU サイクルが発生します。これらの操作をインテル® DSA にオフロードします。インテル® DSA インスタンスが `accel-config` などのツールにより列挙および構成されている場合、プログラマーはまず記述子のデータ構造を割り当ててフィルする必要があります（**図 3**）。記述子には、操作のタイプ、ソースアドレスとデスティネーション・アドレス、データ長など、目的の操作に関する情報が含まれています。

```
struct dsa_completion_record comp __attribute__((aligned(32)));

struct dsa_hw_desc desc = { };

desc.opcode = DSA_OPCODE_MEMMOVE;

/*
 * Request a completion - since we poll on status, this flag
 * must be 1 for status to be updated on successful
 * completion
 */
desc.flags = IDXD_OP_FLAG_RCR;

/* CRAV should be 1 since RCR = 1 */
desc.flags |= IDXD_OP_FLAG_CRAV;

/* Hint to direct data writes to CPU cache */
desc.flags |= IDXD_OP_FLAG_CC;

desc.xfer_size = BLEN;
desc.src_addr = (uintptr_t)src;
desc.dst_addr = (uintptr_t)dst;
comp.status = 0;
desc.completion_addr = (uintptr_t)&comp;
```

図 3. 記述子を初期化

記述子の準備ができたら、プログラマーの次のステップは、現在のプログラムで開かれてマップされている WQ に記述子を送信することです。WQ のタイプに応じて、ソフトウェアは ENQCMD または MOVDIR64B 命令を使用して記述子を送信します。どちらの命令も GNU* GCC（バージョン 10 以降）でサポートされています。プログラマーは、x86 組込み関数を使用してこれらの命令を呼び出すことができます（**図 4**）。

```
#include <x86gprintrin.h>

_mm_sfence();

if (dedicated)
    _movdir64b(wq_portal, &desc);
else {
    retry = 0;
    while (_enqcmd(wq_portal, &desc) && retry++ < ENQ_RETRY_MAX);
}
```

図 4. 記述子を送信

記述子の完了を確認するには、プログラマーは、インテル® DSA によって更新される、完了レコードのステータスフィールドをポーリングする必要があります。最も一般的な方法はスピンポーリングです（**図 5**）。PAUSE や UMONITOR/UMWAIT などの新しい命令を適用すると、コアが異なる電力ステートに移行できるため、プロセッサーの消費電力をさらに削減することができます。（詳細は、「インテル® 64 および IA-32 アーキテクチャー・ソフトウェア開発者マニュアル」（英語）を参照してください。）

```
retry = 0;
while (comp.status == 0 && retry++ < COMP_RETRY_MAX);

if (comp.status == DSA_COMP_SUCCESS) {
    /* Successful completion */
} else {
    /* Descriptor failed or timed out
     * See the "Error Codes" section of the Intel® DSA Architecture Specification for
     * error code descriptions
     */
}
```

図 5. 記述子の完了を確認

ソフトウェア・ライブラリーを利用したインテル® DSA の活用

インテル® Data Mover Library（インテル® DML）とインテル® DSA Transparent Offload Library（インテル® DTO）（英語）を利用すると、インテル® DSA が使いやすくなります。インテル® DML は、データの移動と変換を行う高レベルの C/C++ API のセットを提供し、利用可能な場合は対応するインテル® DSA ユニットを呼び出します。インテル® DML は、高度な機能（インテル® DSA のハードウェア操作、非同期オフロード、ロードバランスなど）をサポートしています。インテル® DML v1.0.0 は、GitHub*（英語）から入手できます。リポジトリーのクローンを作成し、必要なパッケージをインストールした後、単純な CMake コマンドでコンパイルしてインストールできます（インストール・オプションの詳細は、インテル® DML のドキュメント（英語）を参照してください）。

Windows* OS:

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=<install_dir> -G "NMake Makefiles" ..
cmake --build . --target install
```

Linux* OS:

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=<install_dir> ..
cmake --build . --target install
```

図 6 に、インテル® DML の基本的な使用法を示します。基本的な手順は変わりませんが、詳細はインテル® DML API と抽象化により隠されています。キー値格納アプリケーションの例を続けます。記述子を手動で準備して送信する代わりに、プログラマーはインテル® DML を呼び出して、指定されたメモリーコピーをインテル® DSA にオフロードできます。

```
#include <dml/dml.hpp>
#include <numeric>
#include <vector>
#include <iostream>

constexpr auto size = 1024u;  // 1 KB

template <typename path>
int execute_mem_move() {
    std::cout << "Starting dml::mem_move example..." << std::endl;
    std::cout << "Copy 1KB of data from source into destination..." << std::endl;

    // Prepare data
    auto src = std::vector<std::uint8_t>(size);
    std::iota(src.begin(), src.end(), 0u);
    auto dst = std::vector<std::uint8_t>(size, 0u);

    // Run operation
    auto result = dml::execute<path>(dml::mem_move, dml::make_view(src), dml::make_view(dst));

    // Check result
    if (result.status == dml::status_code::ok) {
        std::cout << "Finished successfully." << std::endl;
    }
    else {
        std::cout << "Failure occurred." << std::endl;
        return -1;
    }

    if (src != dst) {
        std::cout << "Operation result is incorrect" << std::endl;
        return -1;
    }

    return 0;
}
```

図 6. インテル® DML を使用した基本的なデータ移動の例

インテル® DTO は、アプリケーションがソースコードを変更することなくインテル® DSA を透過的に活用できるようにする、干渉の少ないライブラリーです。ユーザーは、`-ldto` および `-laccel-config` リンカーオプションを使用してライブラリーを動的にリンクするか、アプリケーションの再コンパイルが不要な `LD_PRELOAD` の設定でライブラリーを事前にロードできます。`memmove()`、`memcpy()`、`memset()`、または `memcmp()` などの一般的なシステム API 呼び出しが使用されると、それらの呼び出しはインターセプトされ、対応するインテル® DSA の同期操作にアクセスするインテル® DTO の関数に置換されます。

汎用ライブラリーのインテル® DML とインテル® DTO に加えて、インテル® DSA を利用するドメイン固有のソフトウェア・ライブラリーもあります。例えば、ハイパフォーマンスなカーネルバイパス・ネットワーク・プログラミング向けの Data Plane Development Kit（DPDK）では、インテル® DSA がほかのベンダーの DMA エンジンとともに抽象化され、**dmadev** ライブラリーにラップされています。ユーザーは、対応する API を呼び出して、

ネットワーク・パケットのコピーをインテル® DSA インスタンスにオフロードできます。Storage Performance Development Kit（SPDK）も /lib/accel ライブラリーでインテル® DSA をサポートしており、さまざまな操作をオフロードできます。インテル® DSA は、libfabric や MPI などの HPC ライブラリーでも有効になっており、従来の科学技術計算ワークロードだけでなく、新しい分散 ML/AI アプリケーションでも使用できます。

インテル® DSA を最大限に活用する

ストリーミング・データ用のオンチップ・アクセラレーターとして、インテル® DSA は、プログラミング・モデルを適切にチューニングすると最適なパフォーマンスをもたらします。我々の経験に基づいて、インテル® DSA を使用する際のガイドラインと推奨事項をまとめました（詳細は、このテクニカルレポート（英語）を参照してください）。

バッチサイズと転送サイズの適切なバランスを維持する

特定のサイズのワークをインテル® DSA にオフロードするには、メモリー領域全体に対して 1 つの記述子を使用するか、同じ集約サイズに対して複数の小さな記述子を使用してバッチ処理します。一般的な傾向として、同じオフロードされたワークに対して大きなバッチ処理を行うと、スループットは低下します。個々の記述子をインテル® DSA の内部で処理し、対応するデータをメモリーから読み取る必要があるため、増加した記述子を管理するオーバーヘッドが発生し、これらの操作の実効スループットが低下します。オフロードするデータが連続している場合、同じサイズのより大きな 1 つの記述子に結合すると、スループットとレイテンシーの両方が改善される可能性があります。

同期的にオフロードすると、バッチの最大化と転送サイズの最大化の間のスループットの最適点を示すパターンが明らかになります。ワークをある程度の数（4 ～ 8 個）の記述子でバッチ処理すると、最良の結果が得られます。これは、メモリーの連続領域のフェッチとバッチされた記述子の処理によるレイテンシーのバランスが取れるためです。

可能であればインテル® DSA を非同期的に使用する

操作を非同期方式でインテル® DSA にオフロードすると、CPU コアとインテル® DSA ハードウェアの両方に最適な効率とパフォーマンスが提供されます。この方式は、インテル® DML を使用して簡単に実装できます。非同期の可能性が制限されている場合、キャッシュ汚染が許容できるのであれば、4KB 未満の転送サイズには CPU コアを使用します。インテル® DSA は、オンチップ・アクセラレーターとして、キャッシュおよび（ヘテロジニアス）メモリー階層を操作する多くのオプションを提供します。

データの転送先を適切に制御する

常に LLC に送られる完了レコードとは異なり、記述子のデスティネーション・アドレスに書き込まれるデータは、LLC またはメインメモリーに送ることができます。インテル® DSA は、ユーザーが優先するデスティネーションを示すヒントを提供（つまり、ワーク記述子のキャッシュ制御フラグを設定）できるようにすることで、このキャッシュ制御機能の利用を支援します。フラグを 0 に設定すると、LLC に対応するキャッシュラインがある場合は無効にされ、データはメモリーに書き込まれます。フラグを 1 に設定すると、対応するキャッシュラインを割り当てて、

データは直接 LLC に書き込まれます。この機能の基礎となる原理は、インテル® データ・ダイレクト I/O テクノロジー（インテル® DDIO）の原理である、LLC をプロセッサーと I/O デバイス間の中間バッファーとして利用するダイレクト・キャッシュ・アクセス（DCA）スキームと同じです。一般に、インテル® DDIO は、データアクセスのレイテンシーを軽減し、メモリー帯域幅への負担を減らすことにより、システムのパフォーマンスを向上させます。ただし、LLC で干渉を引き起こす可能性があるため、この機能を使用する場合は注意してください。

キャッシュ汚染は、限られたハードウェア・リソースを共有するプロセスに悪影響を及ぼします。多くのデータセンターのワークロードでは、対立するバックグラウンド・キャッシュの退避により生じるレイテンシーによって、プライマリー・アプリケーションに設定された競争力の高いサービス・レベル・アグリーメント（SLA）が損なわれる可能性があります。一方、パフォーマンスにとって重要なデータ、または近い将来コアで使用されるデータをキャッシュに直接書き込むと、アプリケーションのアクセス・レイテンシーとスループットの利点が得られます。プログラマーは、アプリケーションの動作に基づいてデータの転送先を適切に選択する必要があります。

ヘテロジニアス・メモリー・システムでデータを移動する有力な候補としてのインテル® DSA

NUMA リモートメモリー、パーシステント・メモリー、CXL ベースのメモリーなど、異なるメモリーメディア間でのデータの移動は、最新の階層型メモリーシステムでは一般的です。インテル® DSA は、利用可能なハードウェア・リソースを通じて、高度な構成の柔軟性を提供します。適切な構成を活用することにより、インテル® DSA のハードウェア使用率が最適化され、パフォーマンスが向上します。

処理エンジンレベルの並列処理を活用する

グループで使用される処理エンジンの数が観測される最大スループットに影響を与える可能性があります。グループあたりの処理エンジンの数を増やすと、スループットが向上します。転送サイズが小さいほど優れたパフォーマンスが得られるため、ユーザーはこれらのグループにオフロードするタスクの一般的な転送サイズに注意する必要があります。

WQ 構成を最適化する

バッチ処理または専用 WQ（DWQ）を使用すると、共有 WQ（SWQ）と比較して大きな利点が得られます。共有 WQ がほかの多くのスレッドで使用されない限り、多くの DWQ を使用するように構成を変更することにより、使用率とスループットを向上させることができます。使用するスレッドが少ない場合、SWQ のパフォーマンスは低下する可能性がありますが、WQ の総数よりも多くのスレッドを使用すると、同時実行の管理がハードウェアにオフロードされ、ほかの構成よりもパフォーマンスが向上します。また、グループ内の WQ 間のパフォーマンスを分離するため、WQ を共有または専用に構成できます。インテル® DSA の WQ エントリーは限られているため、1 つの WQ に 32 のエントリーを割り当てると、スループットはほぼ最大になります。

まとめ

インテル® DSA や、第 4 世代インテル® Xeon® スケーラブル・プロセッサーに搭載されているほかのオンチップ・アクセラレーターには、データセンター・コストと総所有コストを削減できる可能性があります。この記事では、インテル® DSA とその基本的な使用方法の概要、このアクセラレーターを最大限に活用するいくつかのガイドラインを説明しました。ソフトウェア・エコシステムの成長に伴って、インテル® DSA が多くのデータセンターのワークロードに採用されると考えています。インテルは、オープンソース・コミュニティーと積極的に協力して、このエコシステムを構築および実現しています。

関連情報（英語）

- インテル® データ・ストリーミング・アクセラレーター（インテル® DSA）の導入
- インテル® データ・ストリーミング・アクセラレーター・アーキテクチャー仕様 [PDF]
- インテル® データ・ストリーミング・アクセラレーター・ユーザー・ガイド [PDF]
- インテル® DSA Performance Micros
- インテル® Data Mover Library（インテル® DML）