実験による DirectX* マルチスレッド・レンダリング・パフォーマンスの理解

この記事は、インテル® デベロッパー・ゾーンに公開されている「Understanding DirectX* Multithreaded Rendering Performance by Experiments」の日本語参考訳です。

概要

ゲームエンジンのレンダラーは、多くの場合、CPU 側でパフォーマンス・ボトルネックが発生します。レンダリング・ステップのマルチスレッド化は、コンテンツの詳細を損なうことなくパフォーマンスの問題に対応できる効果的な方法です。この記事では、高性能マルチコア・プロセッサーと典型的なグラフィックス・ハードウェア上で、DirectX* 11 と DirectX* 12 のマルチスレッド API を評価して、マルチスレッド・レンダリング・パフォーマンスに影響する主な要因を解析して、関連する最適化手法を調査します。

1. はじめに

この 10 年間で、PC プロセッサーの性能は大幅に向上しました。現在の PC ゲーム市場では、4 コア CPU が主流であり、コア数の多い CPU が市場シェアを伸ばしています。この傾向は今後も続くことが予想され、数年後にはゲーマー向けに 6 コア CPU の人気が高まるでしょう。

残念ながら、ほとんどのゲームエンジンのレンダラーはまだシングルスレッドであるため、多くの場合、CPU のパフォーマンスがボトルネックとなり、マルチコア計算リソースを利用してパフォーマンスを向上したり、ビジュアルコンテンツの充実を図ることができません。例えば、多くの可視オブジェクトを使用して大規模な屋外シーンをレンダリングする場合、シングルスレッドのレンダラーでは、ほかのコアはアイドル状態のまま 1 つの CPU コアが全負荷で実行され、プレイ可能なフレームレート以下のパフォーマンスとなることがよくあります。

DirectX* 11 以降では、マルチスレッドでの Direct3D* (D3D) アプリケーション・プログラミング・インターフェイス (API) 呼び出しを正式にサポートしています。DirectX* 11 のマルチスレッド・サポートは、即時コンテキストと遅延コンテキストの 2 種類に対応しています (図 1)。異なるスレッドで異なる遅延コンテキストを同時に使用して、即時コンテキストで実行するコマンドリストを生成できます。このマルチスレッド手法により、複雑なシーンを同時に実行可能なタスクに分割できます [1]。

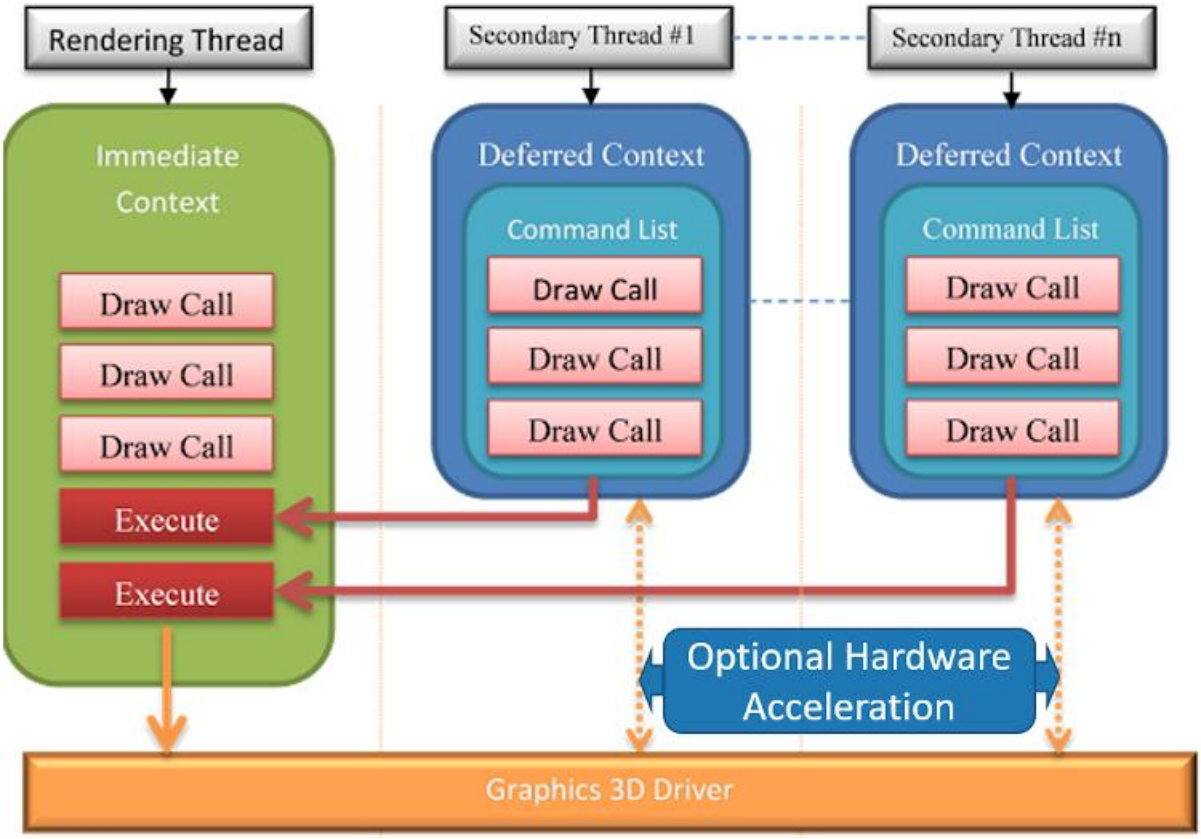


図 1. DirectX* 11 のマルチスレッド・モデル

DirectX* 11 のマルチスレッドは、D3D ランタイムでサポートされますが、ハードウェア・アクセラレーションはオプションです [2]。ハードウェア・アクセラレーションのサポートにより、ドライバー負荷の一部をコマンドリストのビルドと一緒に並列化できます。図 2 は、異なるグラフィックス・デバイスにおけるハードウェア・アクセラレーションを示しています。「Driver Concurrent Creates (ドライバーの同時作成)」はすべてのデバイスでサポートされていますが、「Driver Command Lists (ドライバー・コマンド・リスト)」は NVIDIA* グラフィックスでのみサポートされています。

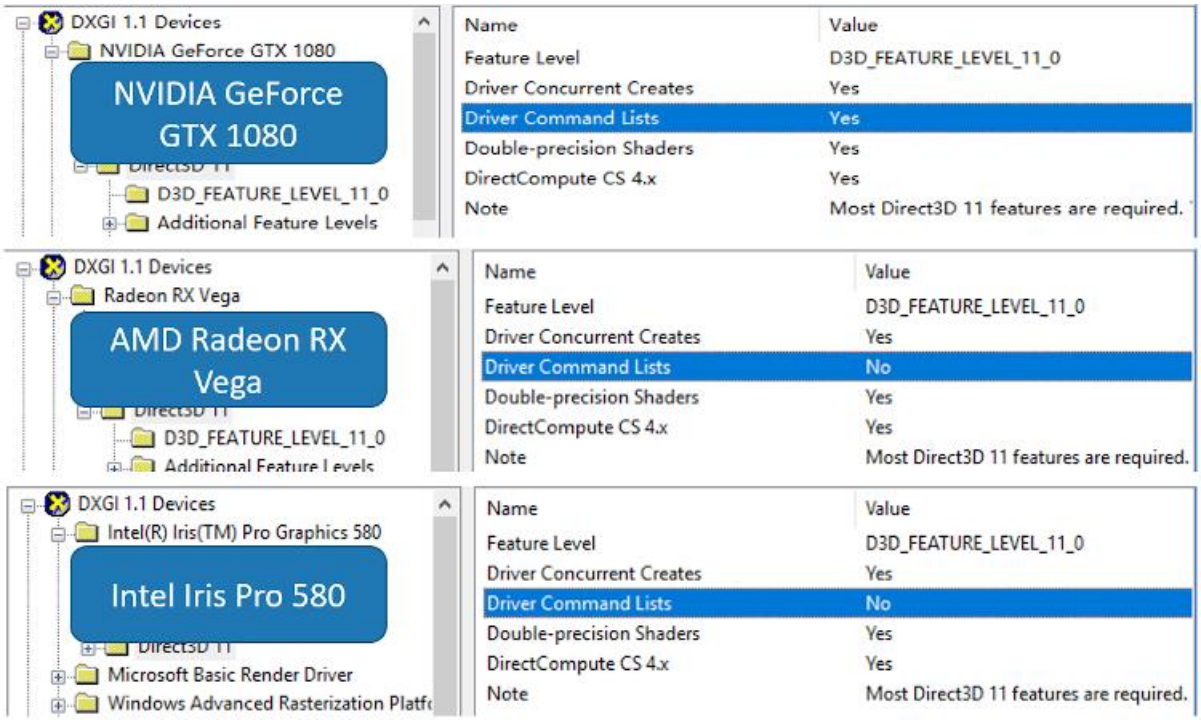
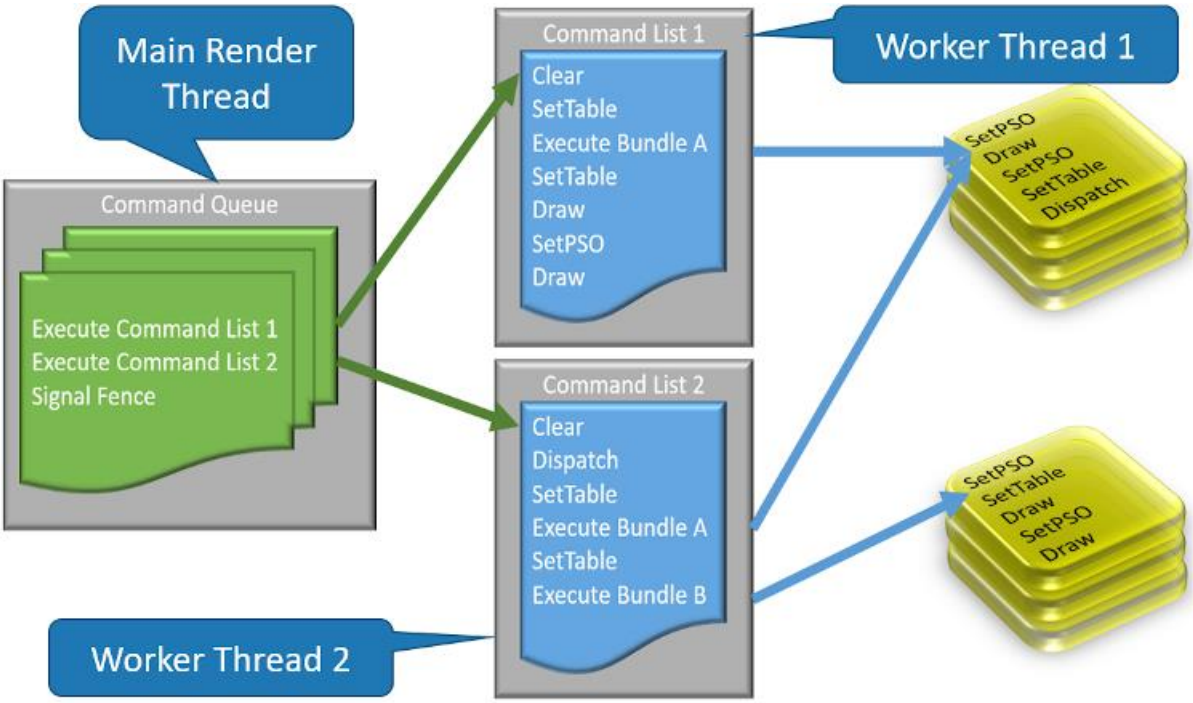


図 2. DirectX* 11 マルチスレッド向けのオプションのハードウェア・アクセラレーション

DirectX* 12 のマルチスレッドは、API 呼び出しのオーバーヘッドを大幅に軽減しています。DirectX* 11 のデバイス・コンテキストの概念を排除して、代わりにコマンドリストを使用して D3D API を呼び出し、コマンドキューを介してコマンドリストを GPU に送信します (図 3)。すべての DirectX* 12 グラフィックス・ハードウェアは、DirectX* 12 マルチスレッド向けのハードウェア・アクセラレーションをサポートします。



*Reference to "Direct3D 12 API Preview" presented by Max McMullen, Microsoft

図 3. DirectX* 12 のマルチスレッド・モデル

DirectX* 11 と DirectX* 12 のマルチスレッド・サポートはどちらも、レンダリングのパフォーマンス・ボトルネックに対応するための魅力的な代替手段です。しかし、レンダラーとマルチスレッド・プログラミングはどちらも複雑であり、誤使用によるパフォーマンス・ペナルティーを回避するため、DirectX* マルチスレッドの特性をよく理解する必要があります。マルチスレッド・レンダリングのパフォーマンスに関して、開発者は多くの場合、実際のメリットとデメリット、マルチコアのスケーラビリティーと関連する主な影響要因、実用的なマルチスレッド手法などについて懸念しています。これらをよく理解することは、後述する実験の目的である、適切に構成されたマルチスレッド・レンダラーの効率良い実装につながります。

2. DirectX* マルチスレッド API のパフォーマンス評価

DirectX* 11 と DirectX* 12 のマルチスレッド API のパフォーマンスの評価に使用するワークロードは、DirectX* の公式サンプルである DirectX* SDK (2010 年 6 月) の「MultithreadedRendering11」(図 4)[3] と DirectX* 12 グラフィックス・サンプルの「D3D12Multithreading」(図 5)[4] です。どちらのサンプルも DirectX* マルチスレッド・レンダリング・メソッドの利点を示すためのものであり、フレームごとに数千の描画呼び出しを使用して同じ「squid room」シーンをレンダリングし、テスト・プラットフォーム上ではともに CPU 依存です。しかし、プログラミングの手法や不具合により、どちらのサンプルもマルチスレッドで DirectX* API を呼び出す実際のパフォーマンスのメリットとデメリットを包括的に示していません。そのため、ここではパフォーマンスを評価する目的で、これらのサンプルを最適化および拡張しています。

図 4. DirectX* SDK (2010 年 6 月) の「MultithreadedRendering11」[3]

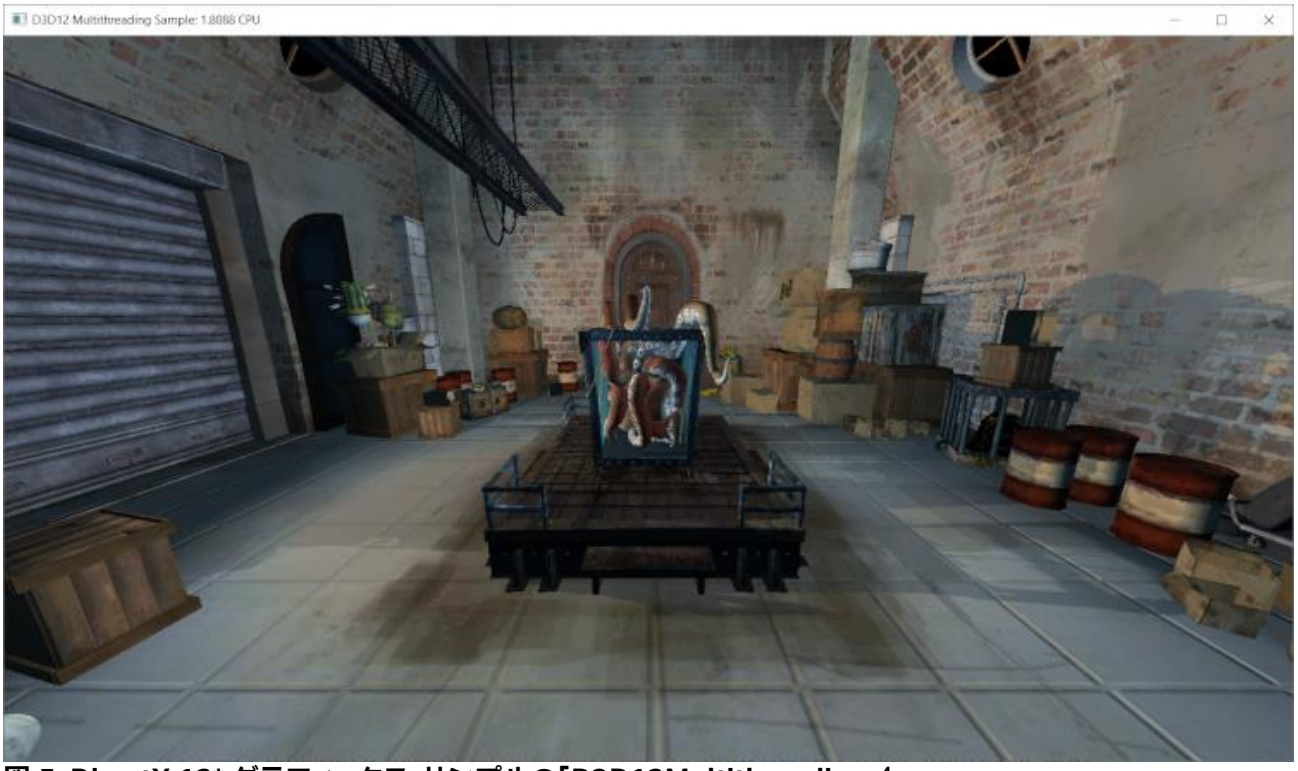


図 5. DirectX 12* グラフィックス・サンプルの「D3D12Multithreading」[4]

この DirectX* マルチスレッド API のパフォーマンス評価は、表 1 に示す異なる構成の 3 つのプラットフォームで実施しました。構成には、主要サプライヤー 3 社のマルチコア CPU とグラフィックス・デバイスが含まれます。GPU ボトルネックを回避するため、使用したグラフィックス・デバイスは、各ブランドの製品の中ではミッドエンドからハイエンドに位置し、すべて DirectX* 11 と DirectX* 12 をサポートしています。DirectX* マルチスレッド・パフォーマンスのスケーラビリティーを測定するため、BIOS によってアクティブな CPU コアを容易に変えられる 10 コアのインテル® Core™ i7-6950X プロセッサーを使用しました。インテル® Core™ i7-6950X CPU にはプロセッサー・グラフィックスが搭載されていないため、統合グラフィックスの評価には 4 コアのインテル® Core™ i7-6770HQ プロセッサーも使用しました。

表 1. テスト・プラットフォーム構成

構成	プラットフォーム A	プラットフォーム B	プラットフォーム C
CPU	インテル® Core™ i7-6950X プロセッサー @ 3.00GHz	インテル® Core™ i7-6950K プロセッサー @ 3.00GHz	インテル® Core™ i7-6770HQ プロセッサー @ 2.60GHz
メモリー	4x8GB RAM	4x8GB RAM	2x8GB RAM
グラフィックス	NVIDIA GeForce GTX* 1080	Radeon RX Vega* 64	インテル® Iris® Pro グラフィックス 580
ドライバー	22.21.13.8494	22.19.677.257	22.20.16.4749
OS	Windows* 10 Enterprise 64 ビット (10.0、ビルド 17134)	Windows* 10 Enterprise 64 ビット (10.0、ビルド 17134)	Windows* 10 Enterprise 64 ビット (10.0、ビルド 17134)

2.1 DirectX* 11 の実験

「MultithreadedRendering11」(図 4) は、DirectX* 11 マルチスレッド・レンダリングのサンプルです。マルチスレッドでの DirectX* 11 API 呼び出しのパフォーマンスとオーバーヘッドを客観的に評価するため、スレッドの同期オーバーヘッドとアプリケーション・レイヤーの負荷を軽減して最適化しています。最適化手法の詳細

は、セクション 3 で説明します。最適化後にリビルドされたワークロード「MTR11_Benchmark」は、表 2 に示すように、フレームごとに 4000 描画呼び出しを含む 5 つのレンダリング・モードを実装します。

表 2. MTR11-Benchmark のレンダリング・モード

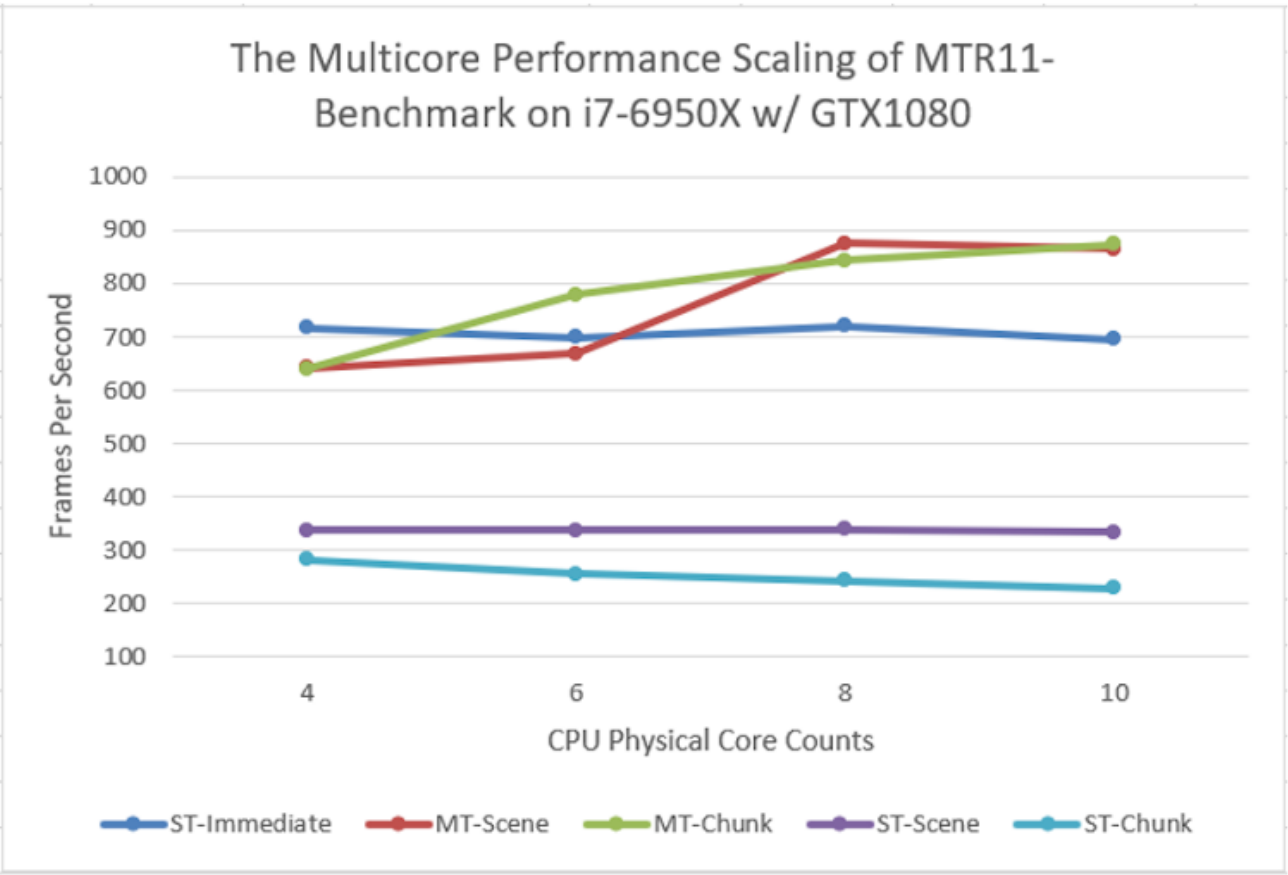
モード	説明
ST-Immediate	シングルスレッド。メインスレッドは、即時コンテキストを使用してすべてのシーンをレンダリングします。
MT-Scene	マルチスレッド。各シーンに遅延コンテキストとそれをレンダリングするワーカースレッドが割り当てられます。メインスレッドは、最終的にすべての遅延コンテキストのコマンドリストを送信します。(フレームごとのコマンドリストの送信数 = シーンの数)
ST-Scene	シングルスレッド。MT-Scene のシリアル化バージョン。メインスレッドは、MT-Scene のワーカースレッドのすべてのタスクをシーケンシャルに実行します。
MT-Chunk	マルチスレッド。各シーンのメッシュは、均等に N (コア数 - 1) チャンクに分割されます。各チャンクに遅延コンテキストとそれを実行するワーカースレッドが割り当てられます。メインスレッドは、各シーンのレンダリングの終了時に、すべての遅延コンテキストのコマンドリストを送信します。(フレームごとのコマンドリストの送信数 = (コア数 - 1) * シーンの数)
ST-Chunk	シングルスレッド。MT-Chunk のシリアル化バージョン。メインスレッドは、MT-Chunk のワーカースレッドのすべてのタスクをシーケンシャルに実行します。

異なる CPU と GPU 構成で実行した MTR11_Benchmark の平均フレームレートを図 6 に示します。このデータから意外なことが分かります。

まず、シングルスレッド・モードの「ST-Immediate」とマルチスレッド・モードの「MT-Scene」および「MT-Chunk」を比較すると、すべてのテスト・プラットフォームにおいて、CPU コア数が少ない場合、シングルスレッドの即時レンダリング (「ST-Immediate」) のほうがマルチスレッドの遅延レンダリング (「MT-Scene」と「MT-Chunk」) よりもパフォーマンスが良いことが分かります。しかし、CPU コア数が多い場合、マルチスレッドの遅延レンダリングのほうがシングルスレッドの即時レンダリングよりもパフォーマンスが、NVIDIA* グラフィックスでは大幅に良く、インテル® グラフィックスではやや良く、AMD グラフィックスでは悪くなります。NVIDIA* グラフィックスは、DirectX* 11 マルチスレッドのハードウェア・アクセラレーションをサポートしていることに注意してください。

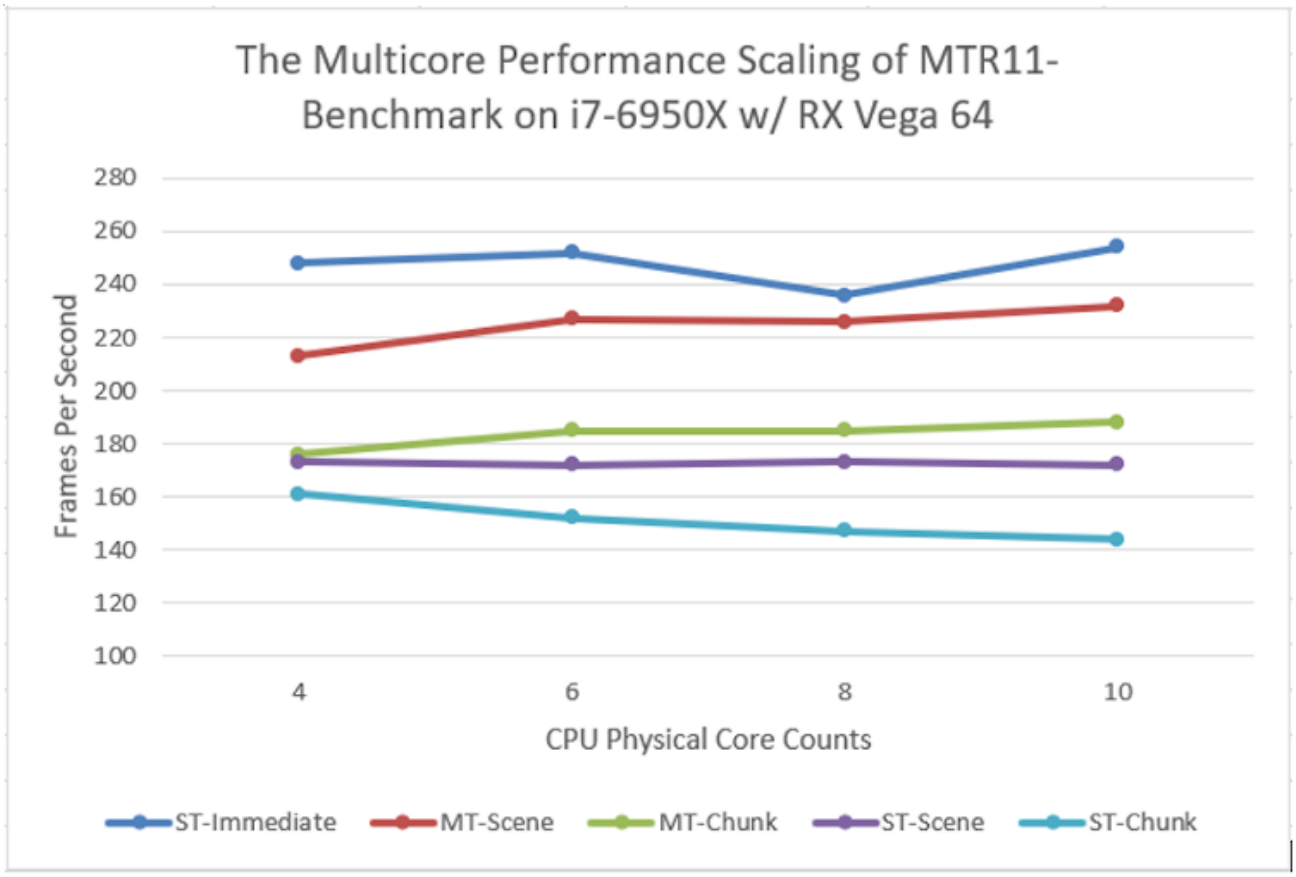
次に、2 つのマルチスレッド・モード「MT-Scene」と「MT-Chunk」を比較すると、すべてのテスト・プラットフォームにおいて、シーンベースのマルチスレッド (「MT-Scene」) のほうが一般にチャンクベースのマルチスレッド (「MT-Chunk」) よりも優れたパフォーマンスを示しています。例外的に、NVIDIA* グラフィックスでは、一部のコア数でチャンクベースのパフォーマンスのほうが優れています (図 6.a)。NVIDIA* グラフィックスは、DirectX* 11 マルチスレッドのハードウェア・アクセラレーションをサポートしており、シーンベースのコマンドリストの送信数はチャンクベースよりも少なくなります (表 2)。

最後に、すべてのシングルスレッド・モード「ST-Immediate」、「ST-Scene」、および「ST-Chunk」を比較して、異なるコンテキストと手法で、マルチスレッドのオーバーヘッドの干渉なしに D3D11 API 呼び出しのコストを評価します。図 6 から、すべてのテスト・プラットフォームにおいて、即時コンテキスト (「ST-Immediate」) での D3D API 呼び出しのほうが遅延コンテキスト (「ST-Scene」と「ST-Chunk」) での呼び出しよりもパフォーマンスがはるかに良いことが分かります。さらに、すべての CPU コア数において、遅延コンテキストでの D3D API 呼び出しは、シーンベース・モード (「ST-Scene」) のほうがチャンクベース・モード (「ST-Chunk」) よりもパフォーマンスが良く、チャンクベース・モード (「ST-Chunk」) は、CPU コア数が増加するとパフォーマンスが低下します。レンダリング・パフォーマンスは、コマンドリストの送信数と負の相関があるように見受けられます。



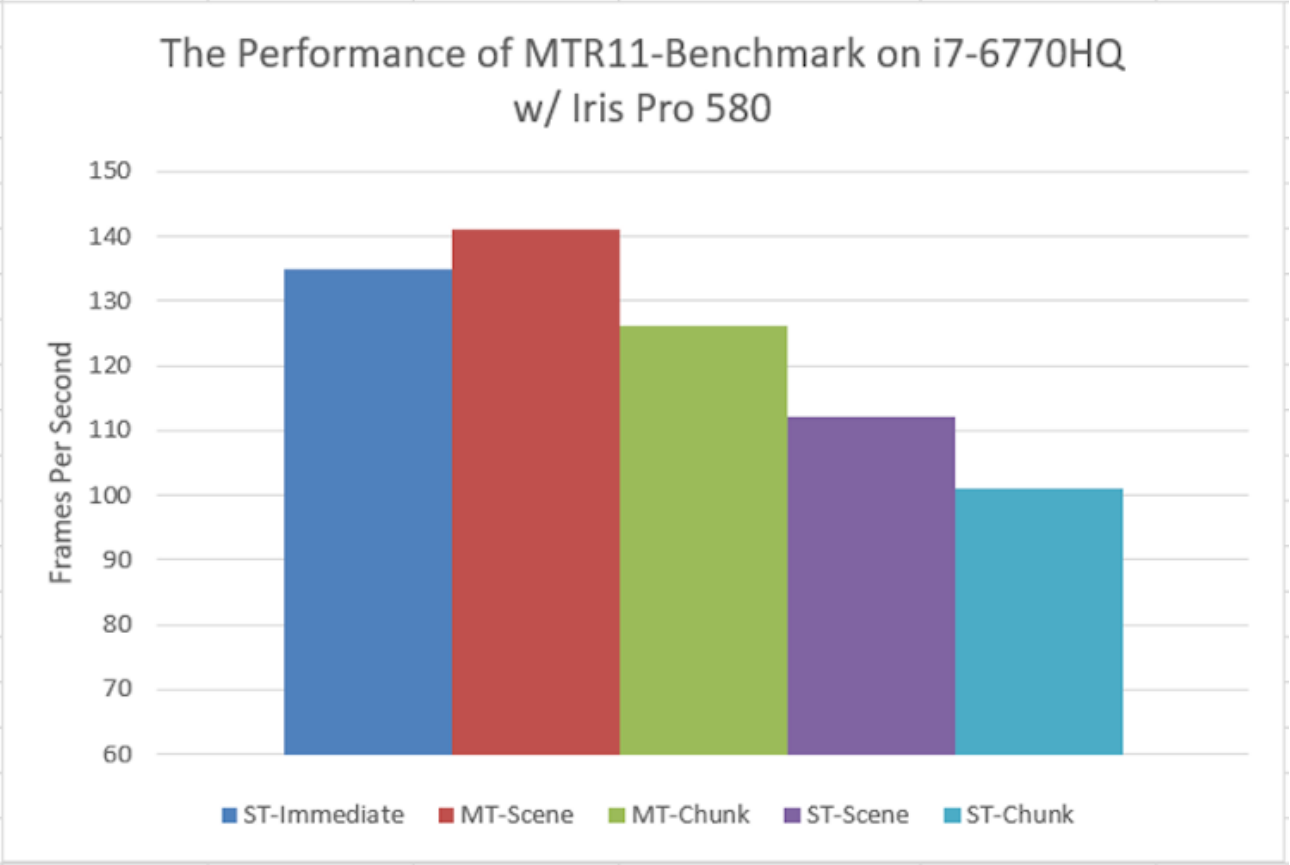
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel® Core™ i7-6950X CPU @ 3.00GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

(6.a) プラットフォーム A のテスト結果



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel® Core™ i7-6950X CPU @ 3.00GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

(6.b) プラットフォーム B のテスト結果



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel Core i7-6770HQ CPU @ 2.60GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

(6.c) プラットフォーム C のテスト結果

図 6. 異なる CPU と GPU 構成での MTR11-Benchmark のパフォーマンス

上記のテスト結果から、DirectX* 11 マルチスレッドのいくつかのパフォーマンス特性を推察できます。

1. DirectX* 11 マルチスレッドには大きなオーバーヘッドが伴います。遅延コンテキストでの D3D API 呼び出しとコマンドリストの送信では、即時コンテキストでの API 呼び出しよりも大きなオーバーヘッドが発生します。また、送信するコマンドリストの数が多くなるほど、オーバーヘッドが増加します。
2. DirectX* 11 マルチスレッドは、特に純粋な D3D API 呼び出しでは、必ずしもパフォーマンスを向上するとは限りません。遅延コンテキストによって生じる大きなオーバーヘッドだけでなく、追加のオーバーヘッドのすべてをマルチスレッドのメリットで相殺できるわけではありません (例えば、シリアル化されたコマンドリストの送信)。そのため、マルチスレッドの遅延レンダリングのほうがシングルスレッドの即時レンダリングよりも合計レンダリング時間が長くなる場合があります。
3. ハードウェア・アクセラレーションは DirectX* 11 のマルチスレッド・パフォーマンスに大きな利点をもたらします。ドライバー負荷は、レンダリング負荷全体の大部分を占めます (図 12)。ハードウェア・アクセラレーションは、ドライバー負荷の一部を並列化して、レンダリング時間を短縮します。

2.2 DirectX* 12 の実験

「D3D12Multithreading」は、DirectX* 12 マルチスレッド・レンダリングのサンプルです。DirectX* 11 の「MultithreadedRendering11」サンプルと比較すると、レンダリング・ロジックが大幅に簡素化されているため、レンダリング処理は DirectX* 12 API を呼び出すだけで済みます。そして、シングルスレッドとマルチスレッドの 2 つのモードのみを実装します。マルチスレッドでの DirectX* 12 API 呼び出しのパフォーマンスとオーバーヘッドを客観的に評価するため、次の機能でサンプルを最適化および拡張しています。

1. アクティブな CPU コア数に応じて、ワーカースレッドの数をスケーリングできるように変更 (オリジナルのサンプルでは 4 に固定)。この変更は、DirectX* 12 マルチスレッドのマルチコア・パフォーマンスのスケーリングの評価に役立ちます。
2. フレームごとの描画呼び出し数を 2000 から「MTR11_Benchmark」と同じ 4000 に増やすため、2 つの重複するシャドウパスを追加。この変更は、高 CPU 負荷でのマルチコア・スケーリング・テストだけでなく、DirectX* 12 と DirectX* 11 のマルチスレッド・パフォーマンスの比較も可能にします。
3. メインスレッドがすべてのコマンドリストの完了を待機してからそれらをバッチ送信するようにして、メインスレッドとワーカースレッド間の同期を最小化。この変更は、DirectX* 12 のマルチスレッド・パフォーマンスに影響する要因の切り分けに役立ちます。
4. 1 つのコマンドリストのみでシングルスレッド・モードを実装。オリジナルのシングルスレッド・モードは、実際には多くのコマンドリストを含むマルチスレッド・モードのシリアル化バージョンであったため、シングルスレッド・モードのパフォーマンスはマルチスレッド・モードを大幅に下回っていました。しかし、実際のシングルスレッド・モードには、1 つまたは少数のコマンドリストのみ必要です。
5. 異なる DirectX* 12 マルチスレッド・メソッドのパフォーマンスを評価するため、5 つのレンダリング・モードを実装 (表 3)。

表 3. MTR12_Benchmark のレンダリング・モード

モード	説明
ST-One	シングルスレッド。メインスレッドは、1 つのコマンドリストのみを使用してすべてのパスをレンダリングします。
MT-Pass	マルチスレッド。各パスにコマンドリストとそれをレンダリングするワーカースレッドが割り当てられます。メインスレッドは、最終的にすべてのコマンドリストをバッチ送信します。(フレームごとのコマンドリストの送信数 = パスの数)
ST-Pass	シングルスレッド。MT-Pass のシリアル化バージョン。メインスレッドは、MT-Pass のワーカースレッドのすべてのタスクをシーケンシャルに実行します。
MT-Chunk	マルチスレッド。各パスで描画されるメッシュは、均等に N (コア数 - 1) チャンクに分割されます。各チャンクにコマンドリストとそれをレンダリングするワーカースレッドが割り当てられます。メインスレッドは、最終的にすべてのパスのコマンドリストをバッチ送信します。(フレームごとのコマンドリストの送信数 = (コア数 - 1) * パスの数)
ST-Chunk	シングルスレッド。MT-Chunk のシリアル化バージョン。メインスレッドは、MT-Chunk のワーカースレッドのすべてのタスクをシーケンシャルに実行します。

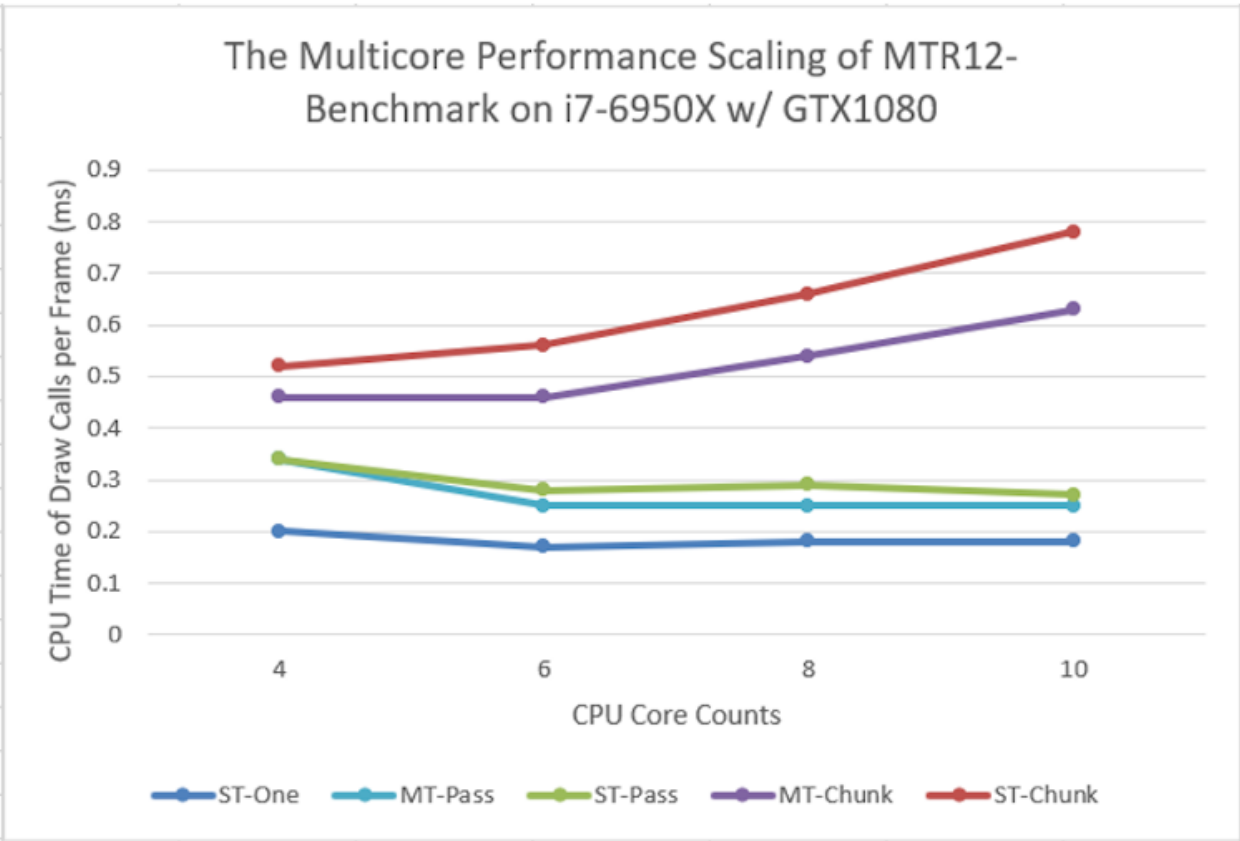
変更後の「D3D12Multithreading」サンプルは「MTR12_Benchmark」です。異なる CPU と GPU 構成で MTR12_Benchmark を実行すると、フレームごとの D3D API 呼び出しの平均 CPU 時間は図 7 のようになり、いくつかの興味深い結果が得られます。

まず、マルチスレッド・モードの「MT-Pass」および「MT-Chunk」とシングルスレッド・モードの「ST-One」を比較すると、ほとんどのテスト・プラットフォームにおいて、1 つのコマンドリストのシングルスレッド・レンダリング (「ST-One」) のほうが一般に複数のコマンドリストのマルチスレッド・レンダリング (「MT-Pass」と「MT-Chunk」) よりもパフォーマンスが優れています。例外的に、AMD* グラフィックスでは、「MT-Pass」のほうが「ST-One」よりもやや高速です (図 7.b)。

次に、2 つのマルチスレッド・モード「MT-Pass」と「MT-Chunk」を比較すると、すべてのテスト・プラットフォームにおいて、パスベースのマルチスレッド (「MT-Pass」) のほうが一般にチャンクベースのマルチスレッド (「MT-Chunk」) よりもパフォーマンスが優れており、チャンクベースのマルチスレッド (「MT-Chunk」) は、CPU コア数が増加するとパフォーマンスが低下します。パスベースのマルチスレッドでは、スレッド数とコマンドリストの送信数が固定されていますが、チャンクベースのマルチスレッドでは、CPU コア数に応じて異なります (表 3)。

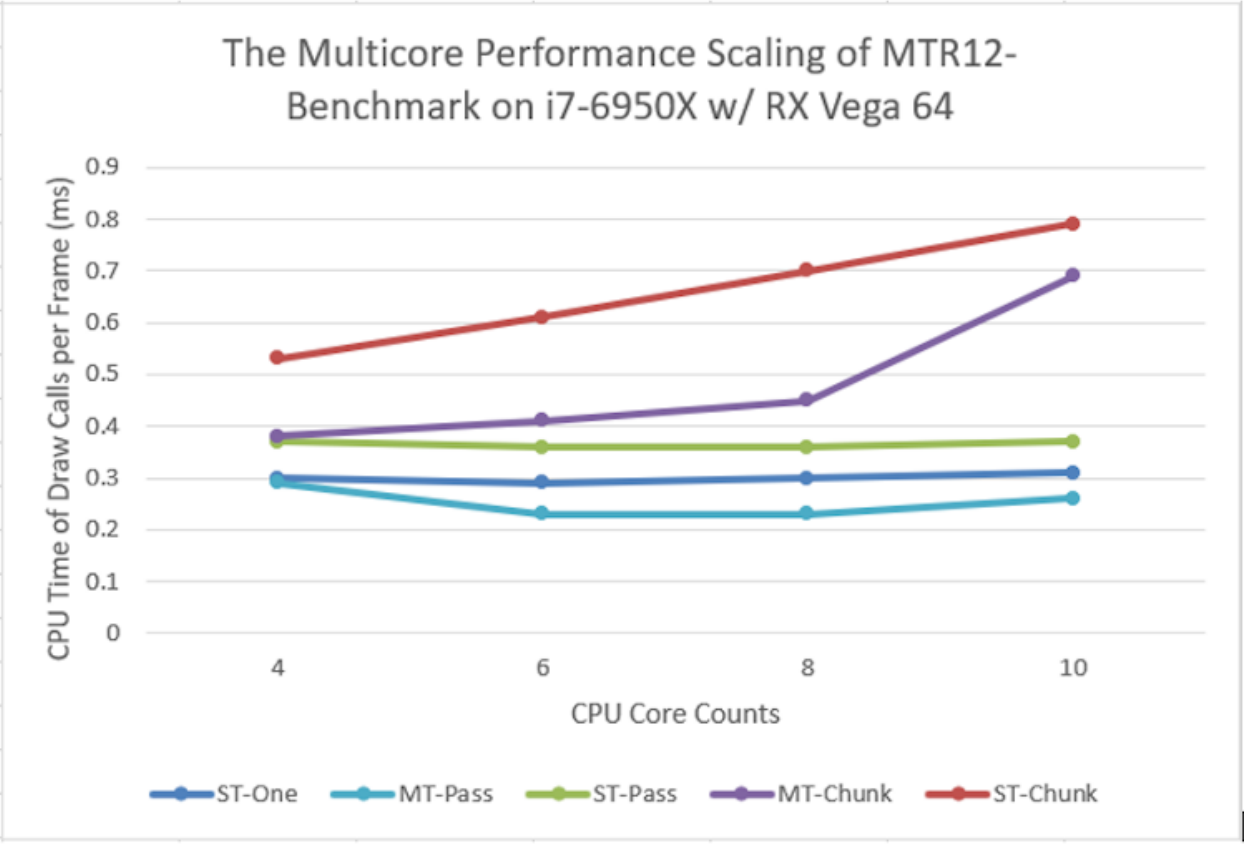
また、すべてのシングルスレッド・モード「ST-One」、「ST-Pass」、および「ST-Chunk」を比較して、異なる手法で、マルチスレッドによるオーバーヘッドの干渉なしに D3D12 API 呼び出しのコストを評価します。図 7 から、すべてのテスト・プラットフォームにおいて、フレームごとに 1 つのコマンドリストを使用した D3D API 呼び出し (「ST-One」) のパフォーマンスが最も優れていることが分かります。フレームごとに少数のコマンドリストを送信するモードの D3D API 呼び出し (「ST-Pass」) は、パフォーマンスが低くなります。また、フレームごとに多数のコマンドリストを送信するモードの D3D API 呼び出し (「ST-Chunk」) は、パフォーマンスが最も低く、CPU コア数が増加するとパフォーマンスがさらに低下します。

最後に、MTR11_Benchmark (図 6) と MTR12_Benchmark (図 7) のテスト結果を比較します。両ワークロードの間に実装の違いはありますが、D3D12 API を使用したレンダリングのほうが、D3D 11 API を使用したレンダリングよりもパフォーマンスが優れているのは明らかです。



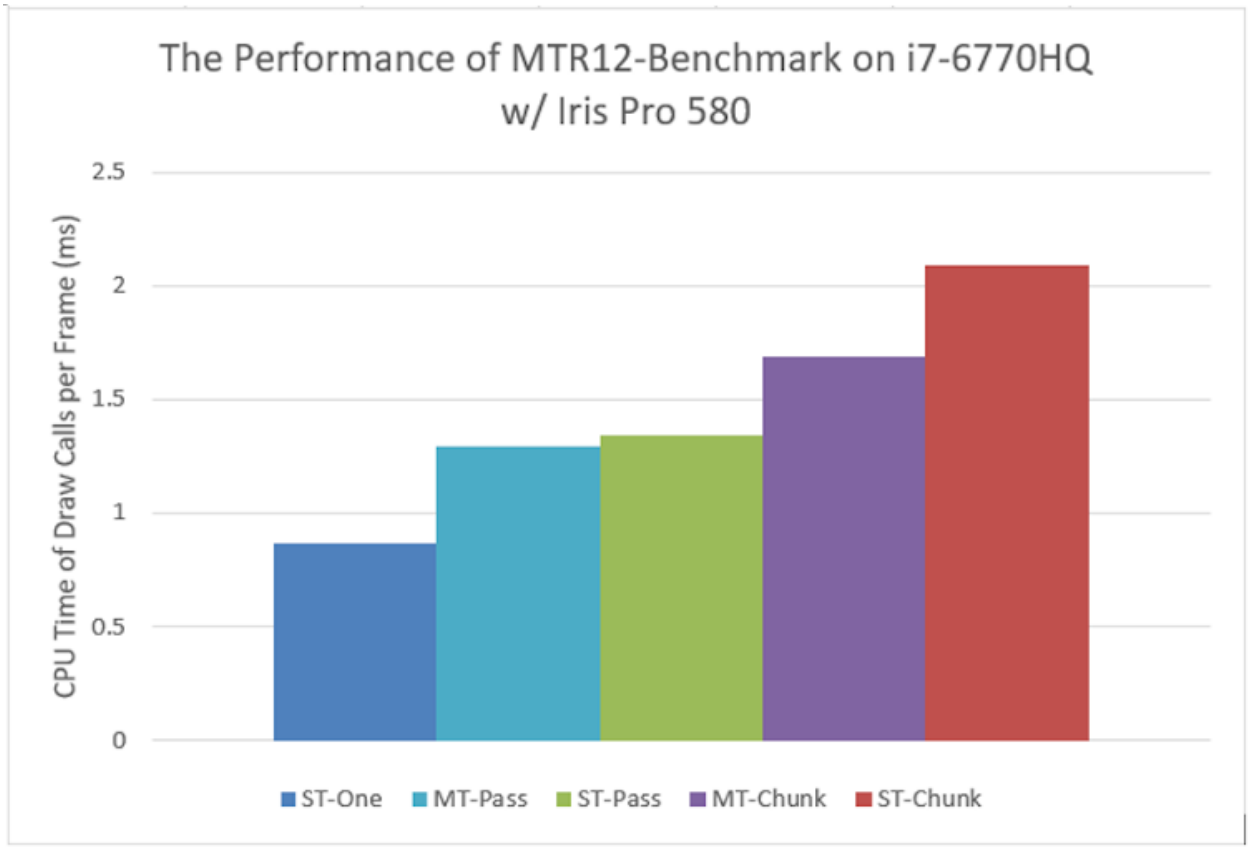
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel® Core™ i7-6950X CPU @ 3.00GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

(7.a) プラットフォーム A のテスト結果



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel® Core™ i7-6950X CPU @ 3.00GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

(7.b) プラットフォーム B のテスト結果



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel Core i7-6770HQ CPU @ 2.60GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

(7.c) プラットフォーム C のテスト結果

図 7. 異なる CPU と GPU 構成での MTR12-Benchmark のパフォーマンス

上記のテスト結果から、DirectX* 12 マルチスレッドには次のパフォーマンス特性があると結論付けることができます。

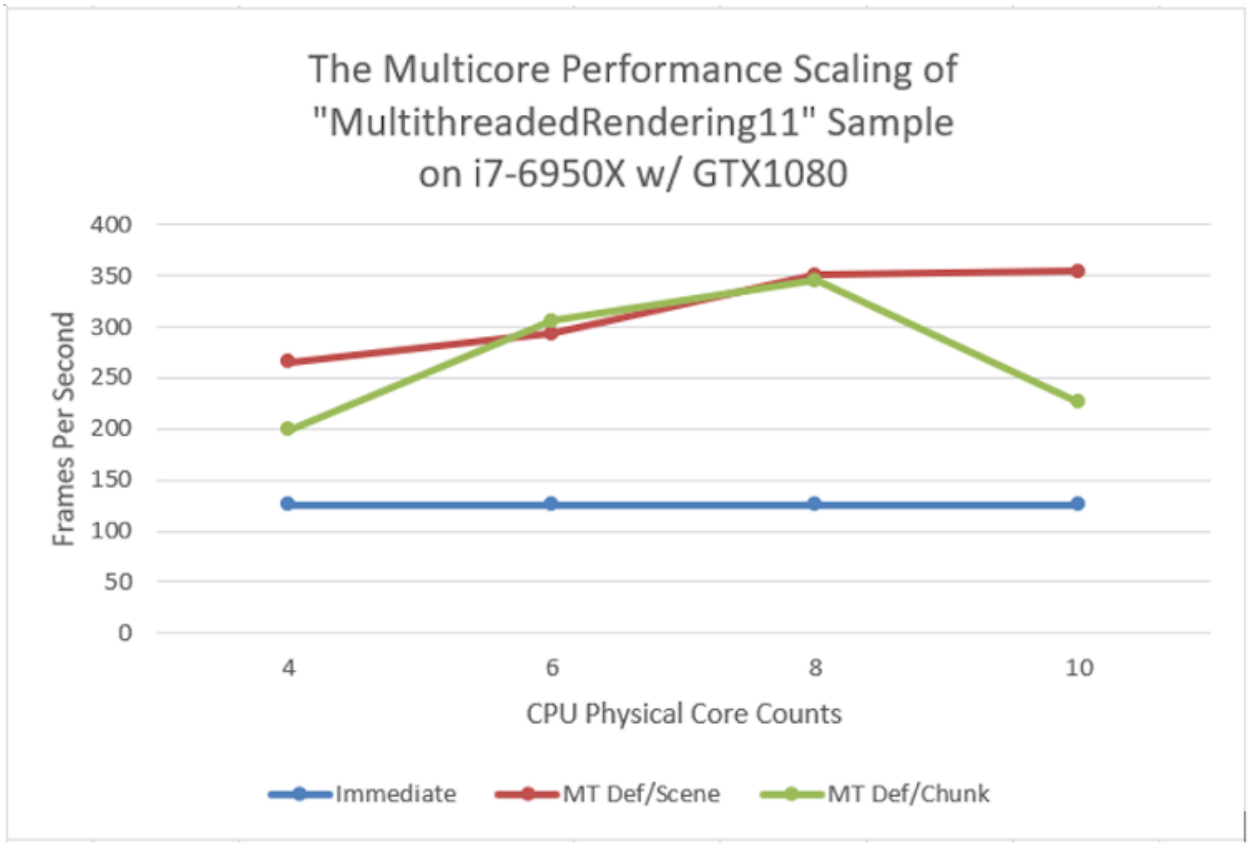
1. D3D12 のコマンドリストの送信は、ほかの D3D12 API 呼び出しよりもはるかに多くのオーバーヘッドを発生します。送信するコマンドリストの数が多くなるほど、オーバーヘッドが大きくなります。
2. DirectX* 12 マルチスレッドは、特に純粋な D3D API 呼び出しでは、必ずしもパフォーマンスを向上するとは限りません。シングルスレッドのレンダリングでは、フレームごとに 1 つまたは少数のコマンドリストを送信するのに対し、マルチスレッドのレンダリングでは、通常、多数のコマンドリストの送信が必要になります。これらの送信はシーケンシャルに行わなければならない可能性があり、コマンドリストを並列に作成することで得られるパフォーマンス・ゲインよりもオーバーヘッドのほうが大きくなることがあります。
3. DirectX* 12 のほうが DirectX* 11 よりもマルチスレッド・レンダリングにより大きなメリットが得られます。DirectX* 12 のマルチスレッドはオーバーヘッドを伴いますが、DirectX* 11 のマルチスレッドのオーバーヘッドと比べるとはるかに少なく、ボトルネックを解消してマルチスレッド・パフォーマンスを向上します。

3. マルチスレッド・レンダリング・パフォーマンスに影響する主な要因

「MultithreadedRendering11」(MTR11) サンプルは、マルチスレッド・レンダリング・パフォーマンスに影響する主な要因を明らかにする良い実験ベースを提供します。セクション 2 で述べた DirectX* のマルチスレッド API のオーバーヘッドに加えて、スレッドの同期オーバーヘッドと並列化可能なアプリケーション・レイヤーの負荷も、マルチスレッド・レンダリング・パフォーマンスに大きく影響します。

3.1 スレッドの同期オーバーヘッド

プラットフォーム A でのオリジナル「MultithreadedRendering11」サンプルのテスト結果 (図 4) は、2 つのマルチスレッド・モード「MT Def/Scene」と「MT Def/Chunk」のパフォーマンスがシングルスレッド・モード「Immediate」のパフォーマンスよりもはるかに優れていることを示しています (図 8)。しかし、CPU コア数が 10 になると、「MT Def/Chunk」モードのフレームレートは大幅に低下します。この異常は調査すべきです。

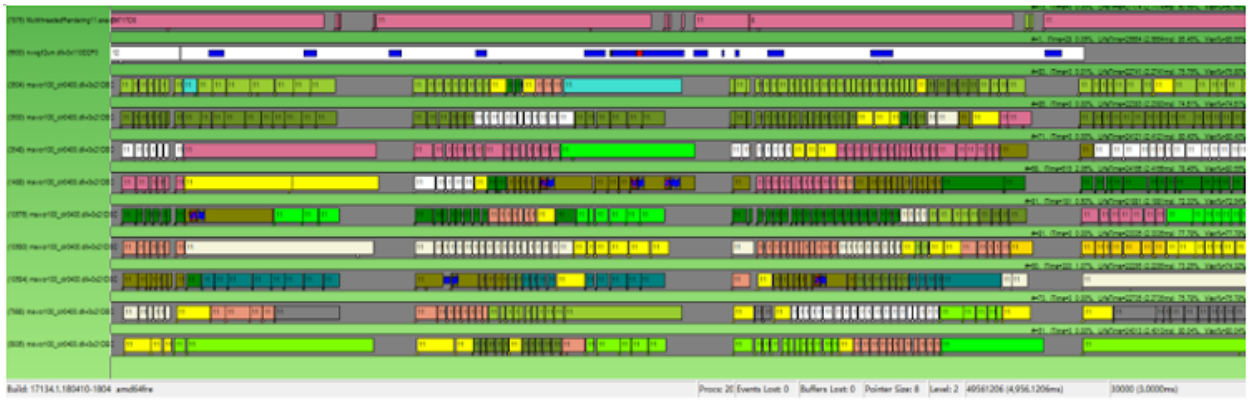


Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel® Core™ i7-6950X CPU @ 3.00GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

図 8. マルチスレッド・レンダリングとシングルスレッド・レンダリングのパフォーマンス比較

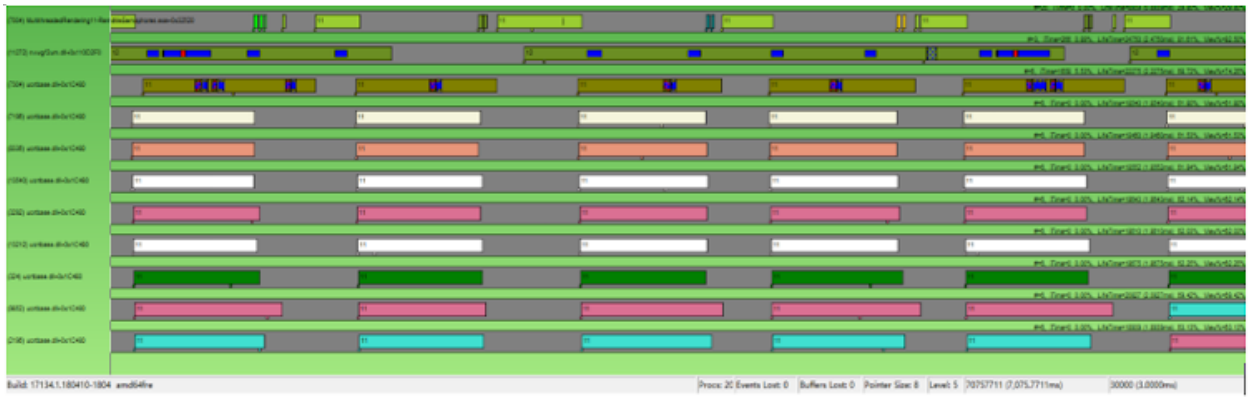
GPUView[9] で 10 コア CPU 上の「MT Def/Chunk」モードのスレッド・アクティビティーを解析すると、ワーカースレッドの実行が頻繁に中断されていることが分かります (図 9.a)。これは、レンダリング中にスレッドの状態変化が多いことを意味します。サンプルのソースコードは、メインスレッドは各シーンをレンダリングする

ため 4000 以上のワークアイテムを継続的に生成し、ワーカースレッドは各ワークアイテムでメインスレッドと同期してそれらを並列に処理していることを示しています。これは、レンダリングにおけるスレッドの同期オーバーヘッドがかなり大きいことを意味します。CPU コア数が増えると、それに対応してワーカースレッドも増えるため、各ワーカースレッドがワークアイテムを待機する可能性と時間も増えます。これにより、スレッドの同期オーバーヘッドがさらに増えて、10 コア CPU ではパフォーマンスが大幅に低下します。



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel® Core™ i7-6950X CPU @ 3.00GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

(9.a) オリジナルの実装

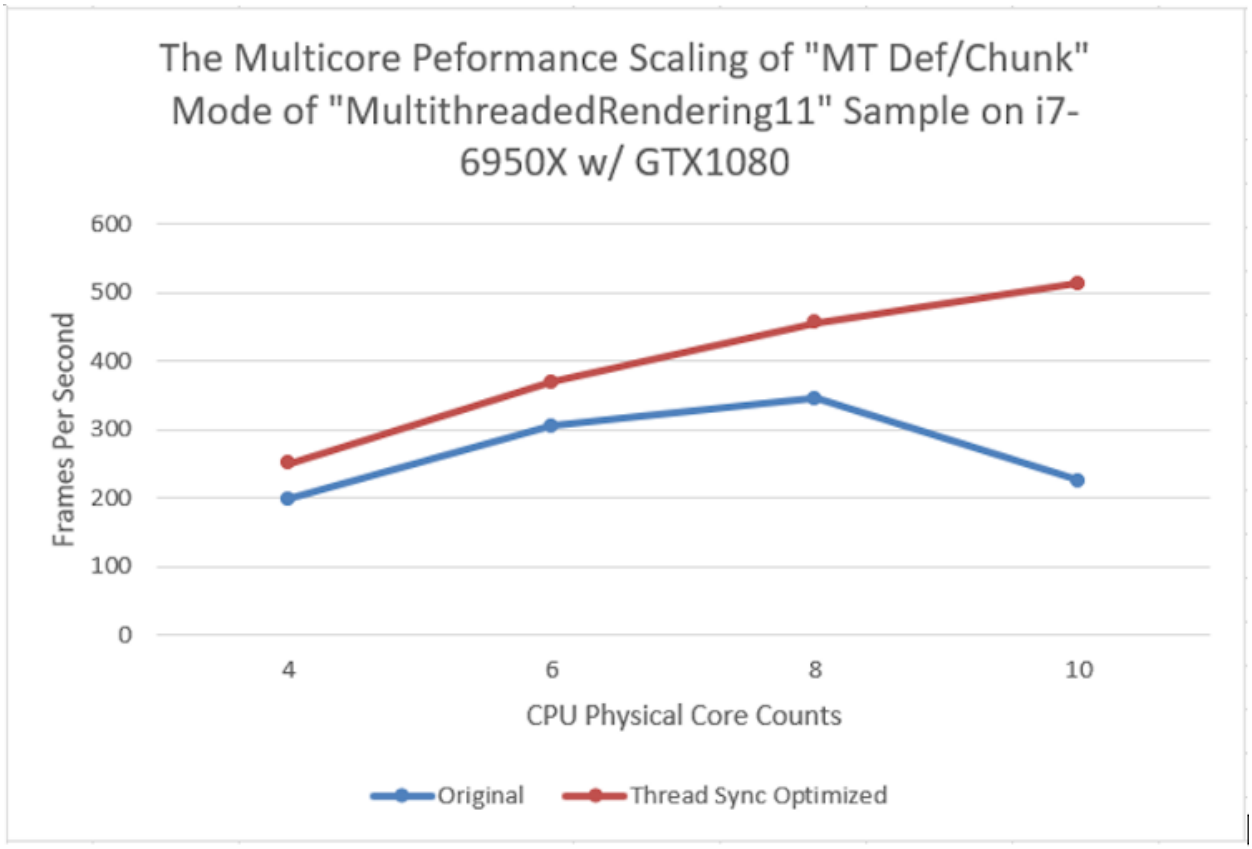


Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel® Core™ i7-6950X CPU @ 3.00GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

(9.b) スレッド同期の最適化された実装

図 9. 10 コア上での「MultithreadedRendering11」サンプルの「MT Def/Chunk」モードのスレッド・アクティビティー

この問題を解決するため、ワーカースレッドがワークアイテムを並列に消費する前にメインスレッドがすべてのワークアイテムを生成するようにサンプルのソースコードを変更しました。これにより、メインスレッドとワーカースレッドの同期の頻度とオーバーヘッドが大幅に軽減されました。図 9.b は、最適化されたサンプルではワーカースレッドの実行の継続性が改善され、各シーンのレンダリング時間が大幅に短縮されたことを示しています。図 10 は、スレッドの同期を最適化したことで、「MT Def/Chunk」モードのパフォーマンスが大幅に向上したことを示しています。これは、10 コアでフレームレートが低下する問題を解決しただけでなく、すべての CPU コア数でこのモードのパフォーマンスを向上しました。



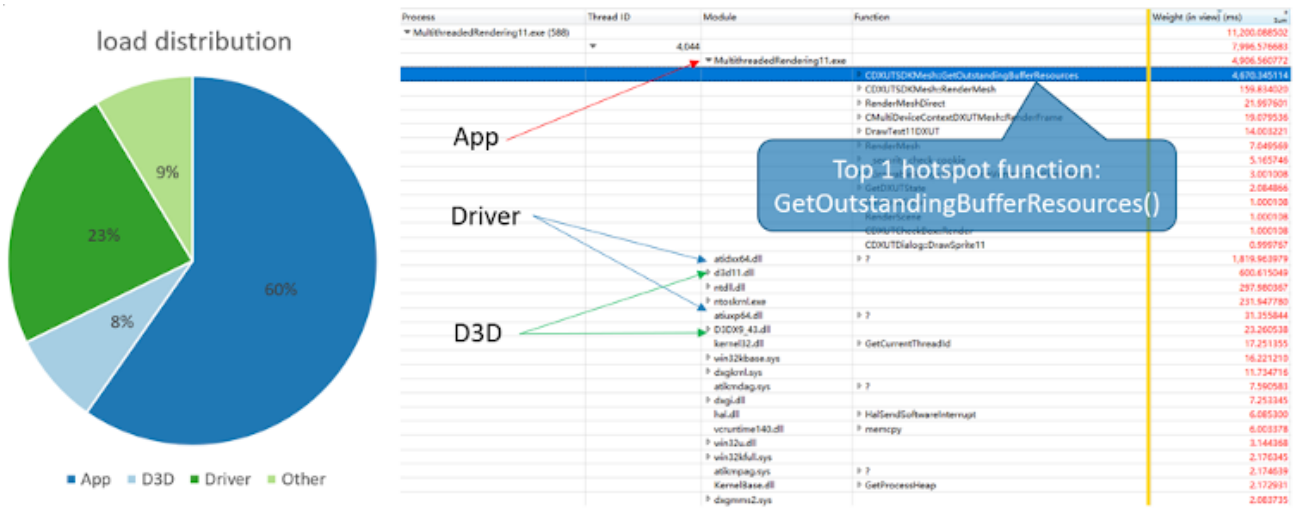
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel® Core™ i7-6950X CPU @ 3.00GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

図 10. スレッドの同期を最適化する前と後の「MultithreadedRendering11」の「MT Def/Chunk」モードのマルチコア・パフォーマンスのスケーリング

この実験は、スレッドの同期オーバーヘッドがマルチスレッド・パフォーマンスに大きく影響することを示しています。不適切なスレッド間の同期は、例えばスレッドのジョブの粒度が細かすぎる場合など、マルチスレッド・レンダリング・パフォーマンスのメリットを損ないます。

3.2 並列化可能なアプリケーション・レイヤーの負荷

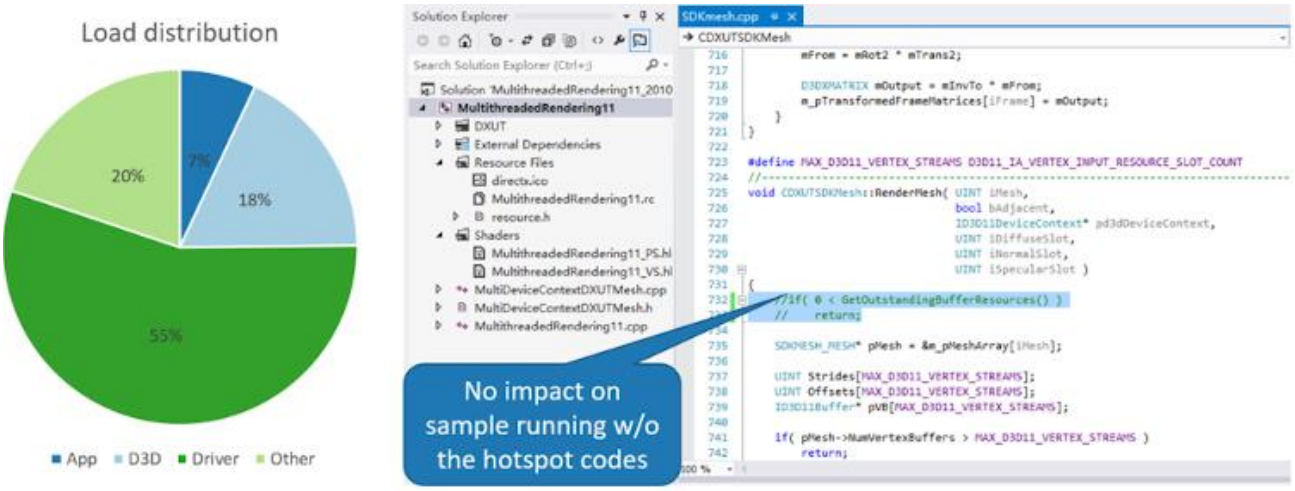
「MultithreadedRendering11」サンプルは、2 つのマルチスレッド・モードのほうがシングルスレッド・モードよりもパフォーマンスがはるかに優れていることを示しています (図 8)。マルチスレッド・レンダリングによりメリットが得られるワークロードの特性を理解するため、サンプルの「Immediate」モードでアプリケーション・ロジック、D3D ランタイム、およびグラフィックス・ドライバーの負荷分布を解析しました (図 4)。Windows* パフォーマンス・アナライザー (WPA)[8] でプロファイルされたモジュールの CPU 利用率の重み (図 11) から、異なるレイヤーの相対負荷を計算できます。それによると、アプリケーション・ロジックの負荷が最も高く (60%)、ドライバーの負荷は 23%、Direct3D* ランタイムの負荷は 8% です。



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel® Core™ i7-6950X CPU @ 3.00GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

図 11.「MultithreadedRendering11」サンプルの「Immediate」モードの負荷分布

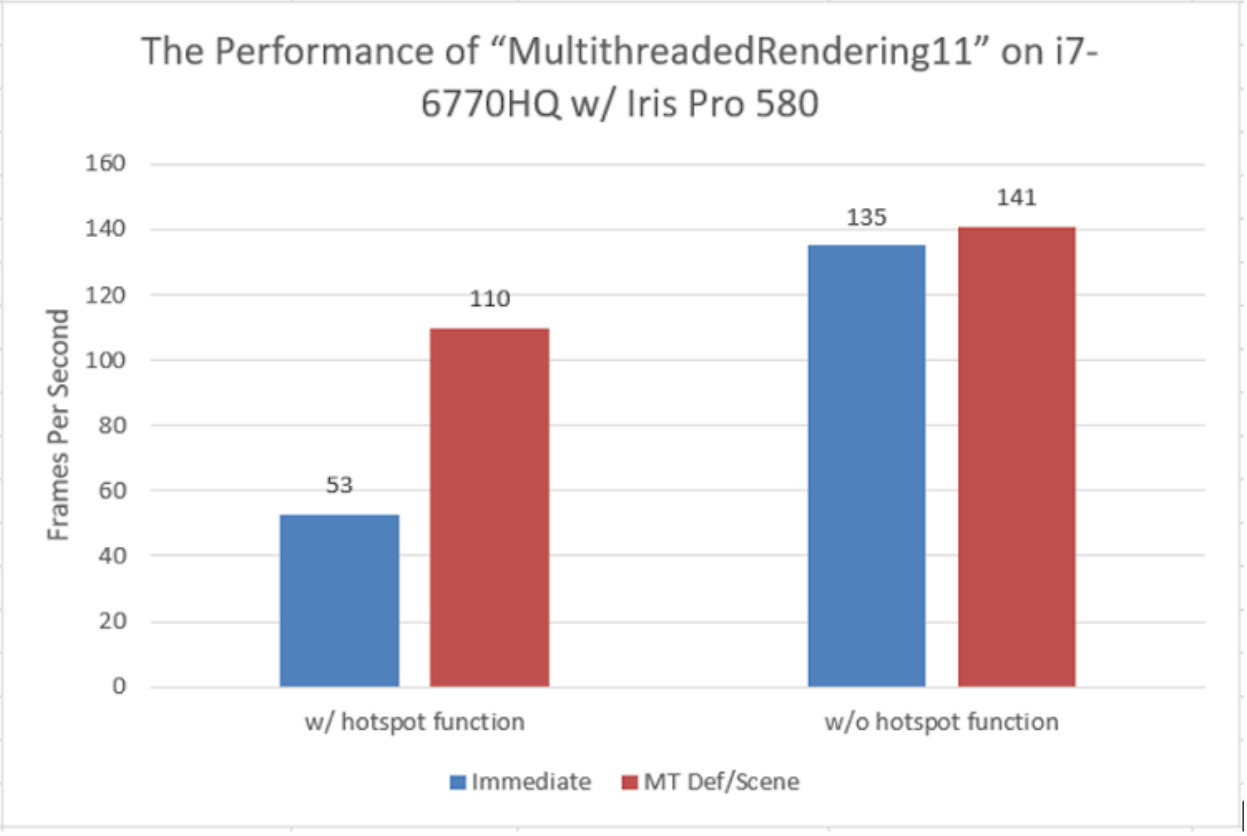
WPA は、最上位のホットスポット関数も明らかにしています (図 11)。ソースコードは、この関数がシーンのすべてのメッシュの頂点バッファーとインデックス・バッファーの検証に使用されることを示しています。奇妙なことに、この関数はメッシュを描画するたびに繰り返し呼び出されていますが、これは明らかに不要です。実際に、このホットスポット関数は、最新の DirectX* SDK サンプルパッケージ [5] に含まれる同名のサンプルで最適化されています。この関数をコメントアウトすると (図 12)、最も高負荷なレイヤーは、アプリケーション・ロジック (7%) からグラフィックス・ドライバー (55%) に変わりますが、プログラムの実行には影響ありません。



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel® Core™ i7-6950X CPU @ 3.00GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

図 12. アプリケーション・レイヤーのホットスポット・コードを削除した後の「MultithreadedRendering11」サンプルの「Immediate」モードの負荷分布

ホットスポット関数をコメントアウトしたサンプルで再度テストすると、シングルスレッド・モード (「Immediate」) と比較したマルチスレッド・モード (「MT Def/Scene」) のスピードアップが大幅に減りました (図 13)。これは、マルチスレッド・レンダリングの大幅なスピードアップの原因が、マルチスレッドでの D3D API ではなく、アプリケーション・レイヤーのホットスポット関数の呼び出しであったことを意味します。



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Performance results are based on testing by Intel engineer as of Jan 15th, 2019 and may not reflect all publicly available security updates. Configuration: Intel Core i7-6770HQ CPU @ 2.60GHz, 32GB RAM, Windows® 10 Enterprise 64-bit (10.0, Build 17134).

図 13. アプリケーション・レイヤーの負荷は、シングルスレッド・レンダリングと比較したマルチスレッド・レンダリングのメリットに大きく影響する

この実験は、並列化可能なアプリケーション・レイヤーの負荷が、マルチスレッド・レンダリングのパフォーマンスに大きく影響することを示しています。並列化可能なアプリケーション・レイヤーの負荷が十分でない場合、マルチスレッド・レンダリングは、シングルスレッド・レンダリングと比較して、パフォーマンスとマルチコア・スケーラビリティーのメリットは得られません。

4. マルチコア・レンダリングのヒントとコツ

上記の実験から得た結論に基づいて、マルチスレッド・レンダリング・パフォーマンスのメリットを得るため、いくつかのヒントとコツを検討することができます。

レンダラーをマルチスレッド化する前に、WPA またはインテル® VTune™ プロファイラー[6] を使用して、レンダラーのアプリケーション・レイヤー、D3D ランタイム、およびグラフィックス・ドライバーの負荷の比率を評価します。アプリケーション・レイヤーが並列化可能であり、その比率が十分に大きい場合、レンダラーをマルチスレッド化することでパフォーマンスが向上します。そうでない場合、メリットがあまり得られないか、パフォーマンスが低下する可能性があります。

レンダラーのマルチスレッド化を設計または選択する場合、開発者は、スレッドの同期とコマンドリストの送信により大きなオーバーヘッドが発生してパフォーマンスが低下する可能性に注意すべきです。そのため、マルチスレッド・レンダリングの実装で、フレームごとのスレッドの同期とコマンドリストの送信数を慎重に制御する必要があります。

マルチスレッド・レンダリングには、「MultithreadedRendering11」サンプルで示したパス (シーン) ベースとチャンクベースの 2 つの基本的な手法があります。パスベースの手法は、パス (シーン) の粒度でレンダリング・タスクを分割します。各フレームで生成されるスレッドのジョブ数は、フレームごとのパス (シーン) の数と等しくなります。そのため、フレームごとのスレッドの同期とコマンドリストの送信数が少なく、マルチスレッドのオーバーヘッドは限定的です。しかし、スレッドの負荷インバランスの可能性や CPU コア数に応じてパフォーマンスがスケーリングしない問題があります。一般に、パスベースの手法は、適度なパフォーマンスの向上をもたらします。

チャンクベースの手法は、パス (シーン) のレンダリング・タスクを小さな粒度のジョブに分割します。各フレームで生成されるスレッドのジョブ数は、CPU コア数の何倍にもなります。理論的に、このメソッドは、スレッドの負荷バランスが良く、マルチコア・パフォーマンスのスケーラビリティーが得られるため、最高のパフォーマンスを達成します。しかし、実際には、特に並列化可能なアプリケーション・レイヤーの負荷が少ない場合、パスベースのメソッドよりも遅くなることがあります。これは、フレームごとのスレッドの同期とコマンドリストの送信が多数生成される可能性があるためです。さらに、CPU コア数が増えると、フレームごとのスレッドの同期とコマンドリストの送信数も増えます。その結果、オーバーヘッドがマルチスレッドのパフォーマンス・ゲインを上回る場合があります。

どちらの手法にも一定の制限があります。マルチスレッド・レンダリングの「理想的な」手法は、アプリケーションの負荷がどれだけ大きくても、そして、DirectX* マルチスレッド向けのハードウェア・アクセラレーションがサポートされているかどうかにかかわらず、すべてのケースでマルチコアを最大限に利用して、パフォーマンス向上が可能であるべきです。

候補となる解決策は、「中間レンダリング・コマンド」の概念を使用して、レンダリング・ロジックとグラフィックス (3D3) API 呼び出しを分離することです。「中間レンダリング・コマンド」は、グラフィックス API の引数をキャッシュするためのラッパーであり、Unreal Engine* 4 の「Rendering Hardware Interface」 (RHI) コマンド [10] に似ています。レンダラーのフロントエンドは、レンダリング・ロジックを実行して「中間レンダリング・コマンド」を生成します。レンダラーのバックエンドは、「中間レンダリング・コマンド」をグラフィックス API に変換して呼び出します。

これにより、レンダリング・ロジックとグラフィックス API 呼び出しはそれぞれ最適なタスク分割手法を使用できます。レンダリング・ロジックは、マルチコアの利用率と負荷バランスを向上する小さな粒度のジョブに分割できます。グラフィックス API 呼び出しは、コマンドリストの送信数を制限するため、フレームごとの描画呼び出し数に応じて、1 つまたはいくつかのジョブ (コマンドリスト) に分割できます。スレッドの同期オーバーヘッドを最小限にするには、インテル® スレッディング・ビルディング・ブロック (インテル® TBB)[7] などのワークスチール・タスク・スケジューラーを備えた効率良いスレッド・ライブラリーの使用を推奨します。

DirectX* マルチスレッドのオーバーヘッドを最小限に抑えることで、レンダラーのアプリケーション・レイヤーの負荷が小さい場合は、広範なグラフィックスカードで、マルチスレッド・レンダリング・パフォーマンスがシングルスレッド・レンダリング・パフォーマンスを下回らないようにします。並列化可能なアプリケーション・レイヤーの負荷が大きい場合は、マルチコアを最大限に利用して、シングルスレッド・レンダリングよりもはるかに優れたパフォーマンスを達成できます。

5. まとめ

この記事では、DirectX* 公式サンプルを変更および拡張したワークロードをベースに一連の実験を行い、DirectX* マルチスレッドのパフォーマンス特性を明らかにしました。

これらの実験を通して、D3D11 と D3D12 のどちらでも、コマンドリストの送信はほかの描画呼び出しよりもオーバーヘッドが大きく、コマンドリストの送信数が多いと描画呼び出しの並列化のパフォーマンス・ゲインが損なわれる可能性があることが分かりました。D3D11 では、ハードウェア・アクセラレーションによりマルチスレッド・レンダリング・パフォーマンスが大幅に向上しました。また、スレッドの同期オーバーヘッドと並列化可能なアプリケーション・レイヤーの負荷も、マルチスレッド・レンダリングとシングルスレッド・レンダリングのパフォーマンスの大きく影響することが分かりました。

これらの定性分析に基づいて、レンダラーをマルチスレッド化する前に並列化可能なアプリケーション・レイヤーの負荷を評価し、マルチスレッド実装でスレッドの同期とコマンドリストの送信数を慎重に制御することを推奨します。また、さまざまな状況でマルチスレッド・パフォーマンスのメリットを維持し、マルチコアを最大限に利用して最高のパフォーマンスを達成するため、「中間レンダリング・コマンド」ベースのマルチスレッド・レンダリング手法の採用を提案します。

CPU のコア数は増加しています。DirectX* のマルチスレッド・パフォーマンスを深く理解してレンダラーをマルチスレッド化することで、マルチコアで CPU ボトルネックを解消し、よりリッチなビジュアルコンテンツを含むシーンをレンダリングするパフォーマンスの余地を確保できる可能性が高まります。

参考資料

1. Direct3D* 11 のマルチスレッディングについて (英語)
2. ハウツー: ドライバーサポートのチェック (英語)
3. DirectX* SDK (2010 年 6 月) (英語)
4. DirectX* 12 グラフィックス・サンプル (英語)
5. Windows* 8.x SDK または Windows* 10 SDK 用 DirectX* SDK サンプル (英語)
6. インテル® VTune™ プロファイラー
7. インテル® スレッディング・ビルディング・ブロック (インテル® TBB)
8. Windows* パフォーマンス・アナライザー
9. GPUView (英語)
10. Render Hardware Interface (RHI)

著者紹介

Sheng Guo は、インテル コーポレーションのゲーム・イネーブリング担当のシニア・アプリケーション・エンジニアです。10 年以上にわたって、主要ゲーム ISV がインテルのプラットフォームとテクノロジーを使用してゲームを改善するのを支援してきました。ゲームのパフォーマンス・プロファイル、最適化、最先端機能のプログラミングなどを専門としています。ゲーム・コミュニティーに多数の技術論文やホワイトペーパーを寄稿しており、学会や学術誌にもいくつかの論文を発表しています。

製品とパフォーマンス情報

[1] 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® 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。
注意事項の改訂 #20110804