VR アプリケーションのパフォーマンス・チューニング

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「Introduction to VR Application Performance Tuning」の日本語参考訳です。

はじめに

この記事では、バーチャル・リアリティー (VR) アプリケーションの開発者に、VR アプリケーション・プロジェクトに適用できる基本的な解析手法および最適化手法を紹介します。この記事は、包括的なリファレンスや既存のガイドを置き換えるものではありません。VR アプリケーションのパフォーマンス解析および最適化のクイックスタート・チュートリアルとして、必要な結果をすばやく得るためにエンジニアが詳しく調査したレシピのサンプルを紹介するものです。

次のガイドとしてご利用ください。

- フルスタック解析。従来のアプリケーションよりも複雑なパフォーマンスの依存関係を調査するためにインテルのエンジニアが使用したツールと手法を紹介します。
- VR アプリケーション・チューニングのレシピ。必要な結果をすばやく得ることができるステップ・バイ・ステップのチュートリアルです。以下の作業を行います。
    - パフォーマンス問題の検出。
    - 根本的な原因の特定。
    - 問題の解決。
- ソリューション・シナリオ。実際に発生したユーザーの問題を解決したときにインテルのパフォーマンス・エンジニアが遭遇したシナリオを紹介します。
- 関連情報。その他の情報。

この記事では Windows* のツールと手法を取り上げますが、多くの手法は Linux* や macOS* の VR アプリケーションでも利用できます。VR 開発環境の急速な変化に伴う新しい情報を反映するため、この記事は今後も更新していく予定です。皆様からの意見やフィードバックをお待ちしています。

フルスタック解析ツールと手法

VR アプリケーションが複雑になるにつれて、最新のパフォーマンス解析ツールと手法が重要になります。1 つのツールで全体を把握することは難しいため、アプリケーションのパフォーマンス特性を包括的に理解するに

は複数のツールと手法を使用する必要があります。この記事では、さまざまな VR ソフトウェア開発キット (SDK) および開発フレームワークの中から、よく使用されるツールを選びました。しかし、この記事で説明する解析タイプには、ほかにも多くのツールが用意されています。

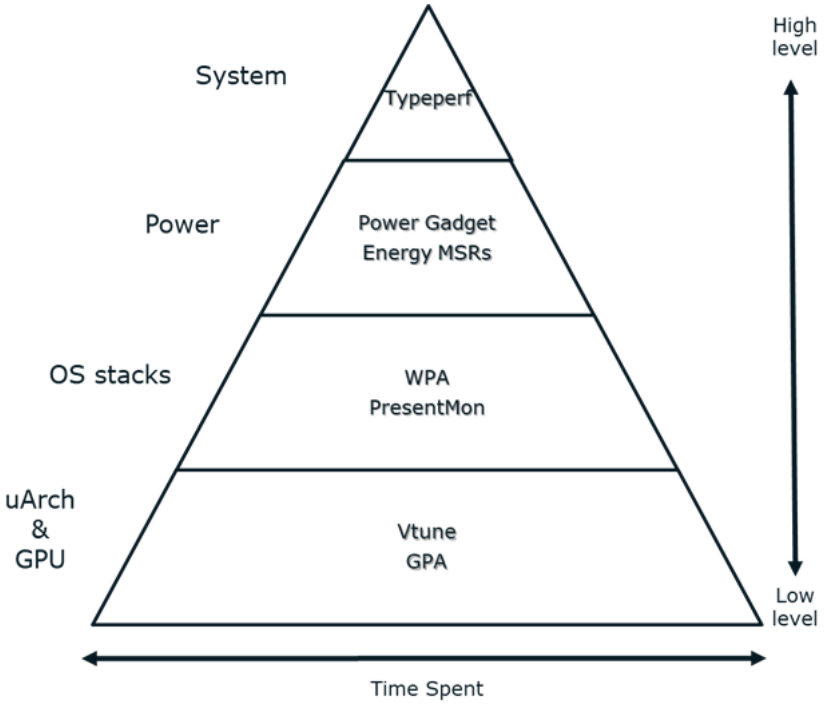


図 1. 解析の構造

図 1 は、ソフトウェア・スタックのさまざまなパフォーマンス解析レベルを示しています。システムレベル (高レベル) の解析で費やされる時間はわずかで、ほとんどの時間はマイクロアーキテクチャー (uArch) レベル (低レベル) の解析で費やされます。費やされる時間は収集されるデータが複雑になるとともに増加し、そのデータの解析に必要な時間も増えます。一般にどのようなツールでもボトルネックの原因は示されますが、すべての解析レベルを調査する必要があるとは限らないことに注意してください。

解析レベル

システム

システムレベルでは、最適化方針を適切に判断する一般的なメトリックが必要です。Windows* の Typeperf などのツールを利用して VR アプリケーションのチューニング・レシピ・セクションを確認し、迅速かつ容易にシステムのアプリケーション負荷を特徴付けることができます。この特徴付けは、直ちに明らかな問題の特定に役立ちます。TypePerf の主なメトリックと説明は、表 1 を参照してください。

電力

電力消費に注目する場合は、さまざまなツールを使用してワークロード全体で消費される電力を測定します。インテル® Power Gadget は、CPU によりレポートされたエネルギーモデル固有レジスター (MSR) を監視し、CPU が消費した電力 (W) を計算して、測定結果を CSV ファイルで出力します。このツールは、CPU 使用率、周波数、温度、インテルの GPU メトリック部分もレポートします。(アプリケーションの消費電力が多い場所を判断するには、CSV ファイルのレポートされたタイムスタンプでワークロードのマーカーを整列します。)

OS スタック

システムを解析してボトルネックを特定したら、OS スタック情報をキャプチャーして、スレッド、モジュール、関数をさらに調べます (シンボルが利用可能である必要があります)。Microsoft* により開発された Windows* パフォーマンス・アナライザー (WPA) は、Windows* パフォーマンス・レコーダー (WPR) によりキャプチャーされたサンプルの後処理を行います。WPA は、グラフィカル・ユーザー・インターフェイスによりグラフおよび表のキャプチャーされたデータを表示し、システム・アクティビティー、計算、ストレージ、メモリー使用量、完全なコールスタックなどのデータを示します (シンボルが利用可能な場合)。

マイクロアーキテクチャーと GPU

CPU で、コールスタックが浅いためにボトルネックを識別できない場合、インテル® VTune™ Amplifier を使用してシリアルおよび並列のボトルネックを識別します。インテル® VTune™ Amplifier は、アルゴリズムを解析し、アプリケーションが利用可能なハードウェア・リソースから利益を得られる場所を理解するのに役立ちます。インテル® VTune™ Amplifier では、ターゲット・アプリケーションを逆アセンブルしたアセンブリー・コードも表示できます。グラフと表でハードウェア・パフォーマンスのカウンターデータを示し、ソースコード (シンボルが利用可能な場合) とアセンブリーに相互リンクできます。インテル® VTune™ Amplifier を使用すると、複数のスレッドにスケーリングしたとき、または大きなスタックの一部として実行したときにアプリケーションのパフォーマンスを制限する、難解なマイクロアーキテクチャーの問題を明らかにすることもできます。

GPU のボトルネックを識別するには、低レベルで GPU のパフォーマンスを理解すべきことがあります。インテル® グラフィックス・パフォーマンス・アナライザー (インテル® GPA) を使用して GPU を詳しく調べ、アプリケーションが GPU 依存か、CPU 依存かを判断します。さらに、インテル® GPA は、グラフィックス・パイプラインの hotspot を特定してフレームレベルでドローコール解析を行い、特定のフレームを詳しく調べて最適化する領域を特定することもできます。

ツール

TypePerf

TypePerf は Windows* に含まれるツールで、コマンドプロンプトから実行できます。1Hz でサンプリングを行い、潜在的な問題の特定に十分な粒度を最小のオーバーヘッドで提供します。多くのワークロードで最初から最後まで実行できます。

一般的なコマンドは次のようになります。

```
typeperf –cf typeperfinput.txt –o workload_perfmon.csv
```

–cf オプションは、メトリックをリストした入力のテキストファイルを指定します。–o オプションは、出力ファイルを指定します。通常、ユーザーは、表計算アプリケーションで生成された CSV ファイルを開き、値をグラフにしてシステムの問題を特定します。

表 1. 主要 TypePerf メトリック

統計	値	メモ
Processor(_Total)\% Processor Time	60.88	
Processor(_Total)\% User Time	38.17	
Processor(_Total)\% Privileged Time	22.71	
Processor(_Total)\% Interrupt Time	0.30	
Processor(_Total)\%DPC Time	0.42	
System\Context Switches/sec	19559.96	コンテキスト・スイッチの値が高い
System\System Calls/sec	1015040.67	システムコールの値が非常に高い!
Processor(_Total)\Interrupts/sec	19417.73	割り込みの値が高い
Memory\Demand Zero Faults/sec	21291.03	
Memory\Page Faults/sec	24319.89	

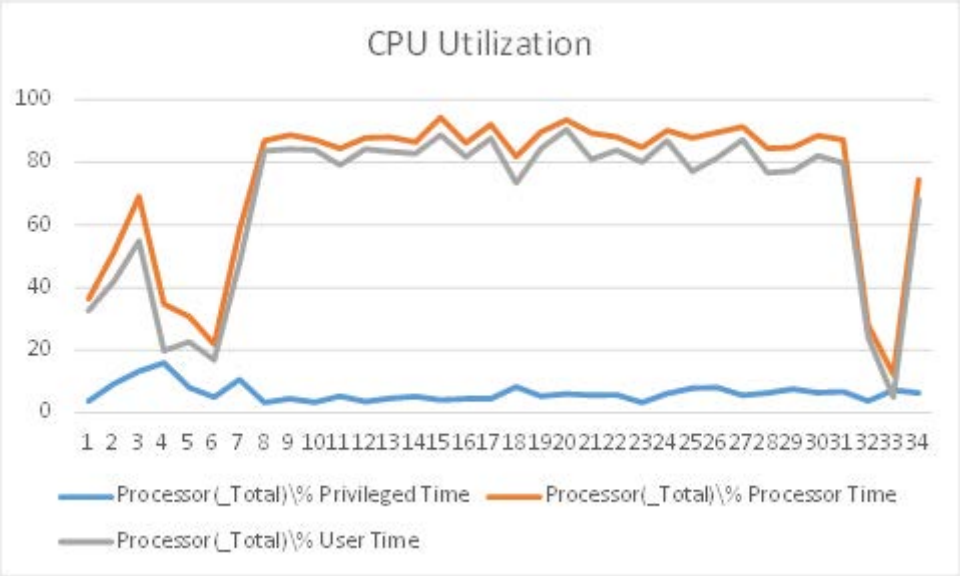


図 2. TypePerf メトリックの典型的なグラフ

このデータを解釈すると状況をすばやく把握できます。下記の表 2 は、TypePerf メトリックのコアセットを調査するガイドラインです。

表 2. 主要 TypePerf メトリック

メトリック	説明	ガイドライン
コンテキスト・スイッチ	論理コアがあるスレッドから別のスレッドに実行を切り替えた場合	アクティブな実行スレッドあたり 10,000/秒未満
システムコール	オペレーティング・システムのサービスの呼び出し	アクティブな実行スレッドあたり 50,000/秒未満
ページフォルト	スレッドが OS の現在のワーキングセットにないページを指した場合	アクティブなスレッドあたり 50,000/秒未満 (大部分はソフトウェア障害)
ハードウェア割り込み	デバイスがタスクを完了したときや注意が必要なときにプロセッサーに行う割り込み	6000 ~ 7000 未満
プロセッサー・キュー	実行する準備ができたキューのスレッド	1 を超えた場合はボトルネックを示す
平均ディスクキュー	サンプリング中に選択されたディスクにキューされた読み取りリクエスト数と書き込みリクエスト数の平均	1 を超えた場合はディスク IO で部分的にゲートされたことを意味し、2 を超えた場合はディスク IO で完全にゲートされたことを意味する
プロセッサー周波数	プロセッサーの周波数	ターボステート (P ステート) を除く

Windows* パフォーマンス アナライザー (WPA)

WPA (図 3) は、Windows* イベント・トレーシング (ETW) ベースのツール、Windows* パフォーマンス・レコーダー (WPR) を利用します。WPR は、WPA を使用して解析できるシステムイベントを記録します。WPR はテスト中のデータを記録し、WPA はキャプチャーされたデータ (CPU、GPU、その他) を表示します。WPA は Windows* アセスメント & デプロイメント・キット (Windows* ADK) に含まれています。

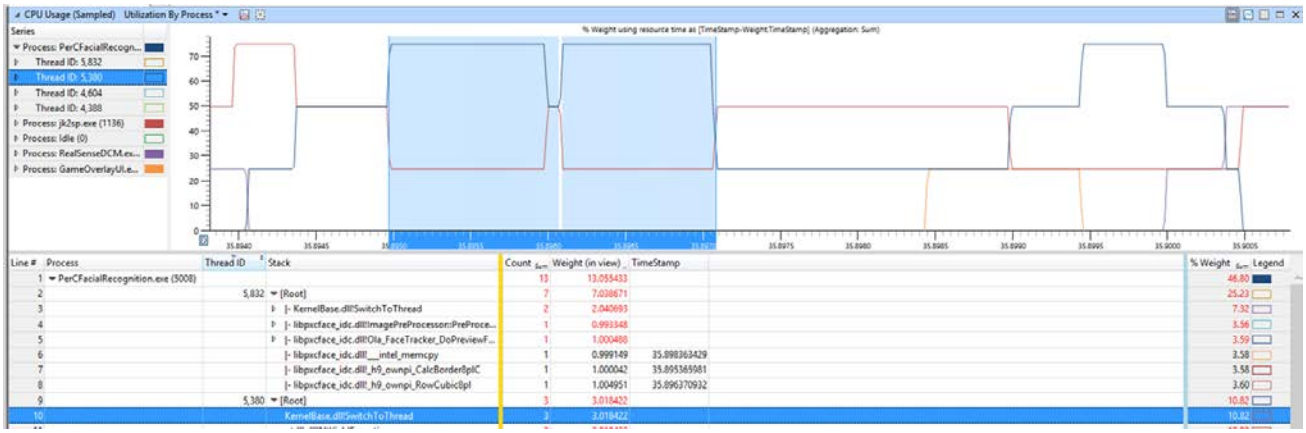


図 3. Windows* パフォーマンス・アナライザーは CPU と GPU の使用状況データを表示

WPR は、Windows* パフォーマンス・ツールキットのフォルダー内の gpuview フォルダーにインストールされる log.cmd ファイルを使用してコマンドラインから実行することもできます。コマンドプロンプトから **log.cmd** を実行して、データ収集を開始および停止します。データ収集を停止した後、トレースのセットが gpuview フォルダーに作成されます。merged.etl は収集対象のファイルで、WPA で開くことができます。WPR、WPA をインストールするには、Windows* アセスメント & デプロイメント・キット (Windows* ADK) を入手します。

インテル® VTune™ Amplifier

インテル® VTune™ Amplifier 2018 (図 4) を使用すると、シリアルおよび並列コードのボトルネックを特定し、コードの実行を高速化できます。このツールを使用してアルゴリズムを解析し、アプリケーションが利用可能なハードウェア・リソースから利益を得られる場所を理解します。インテル® VTune™ Amplifier の評価版は、こちらからダウンロードできます。

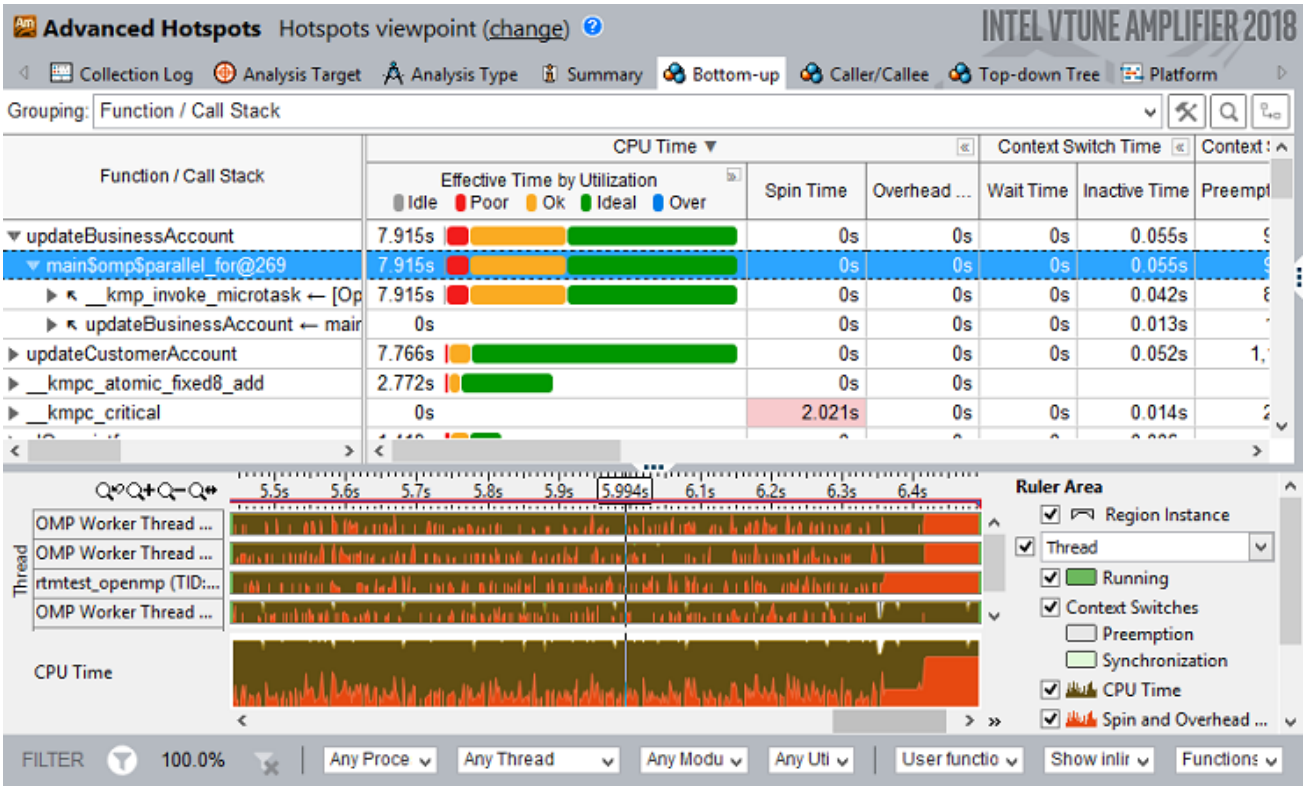


図 4. インテル® VTune™ Amplifier 2018 でコードのボトルネックを特定、解析して最近の CPU のパフォーマンスを最適化

PresentMon

PresentMon を使用して、Windows* のスワップ・チェーン・プレゼンテーションに関連する ETW イベントをトレースします。グラフィックス・アプリケーションの主要パフォーマンス・メトリック (例えば、CPU とディスプレイのフレーム時間およびレイテンシー) をキャプチャーして解析できます。すべてのグラフィックス API で動作し、UWP アプリケーションをサポートします。

PresentMon はインテルにより開発されたオープンソースのツールで、GitHub* (英語) から入手できます。

インテル® Power Gadget

インテル® Power Gadget を使用してインテル® Core™ プロセッサーの消費電力をモニターします。このツールは Windows* および macOS* で動作し、リアルタイムにプロセッサー・パッケージの電力情報をモニターおよび予測するアプリケーション、ドライバー、ライブラリーを含みます。

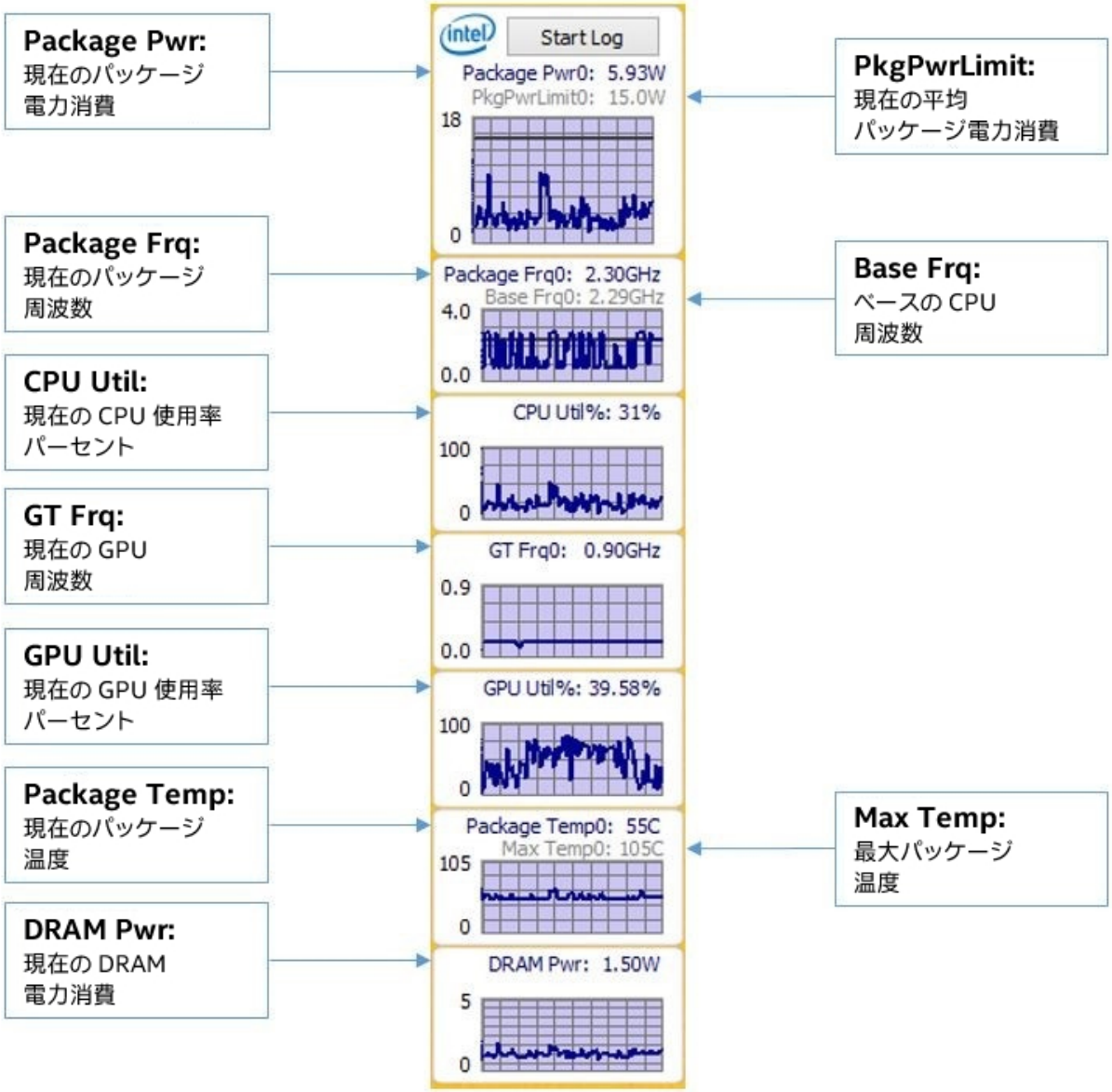


図 5. インテル® Power Gadget はプロセッサーの統合エネルギーカウンターを使用してアプリケーションの電力消費をモニター

インテル® Power Gadget で CSV ログファイルを生成するには、**[Start Log (記録開始)]** ボタンをクリックします。赤の「Rec」が点滅し、記録が開始したことを示します。ワークロードの最後で、**[Stop Log (記録停止)]** をクリックしてデータのキャプチャーを完了します。デフォルトでは、**ログ**はドキュメント・フォルダーに保存されます。

インテル® Power Gadget は、こちら (英語) からダウンロードできます。

インテル® グラフィックス・パフォーマンス・アナライザー (インテル® GPA)

インテル® グラフィックス・パフォーマンス・アナライザー (インテル® GPA) を使用してフレームレベルでボトルネックを特定し、フレームでリアルタイムのテストを行います (図 6)。インテル® グラフィックス・パフォーマンス・アナライザーは、こちら (英語) からダウンロードできます。

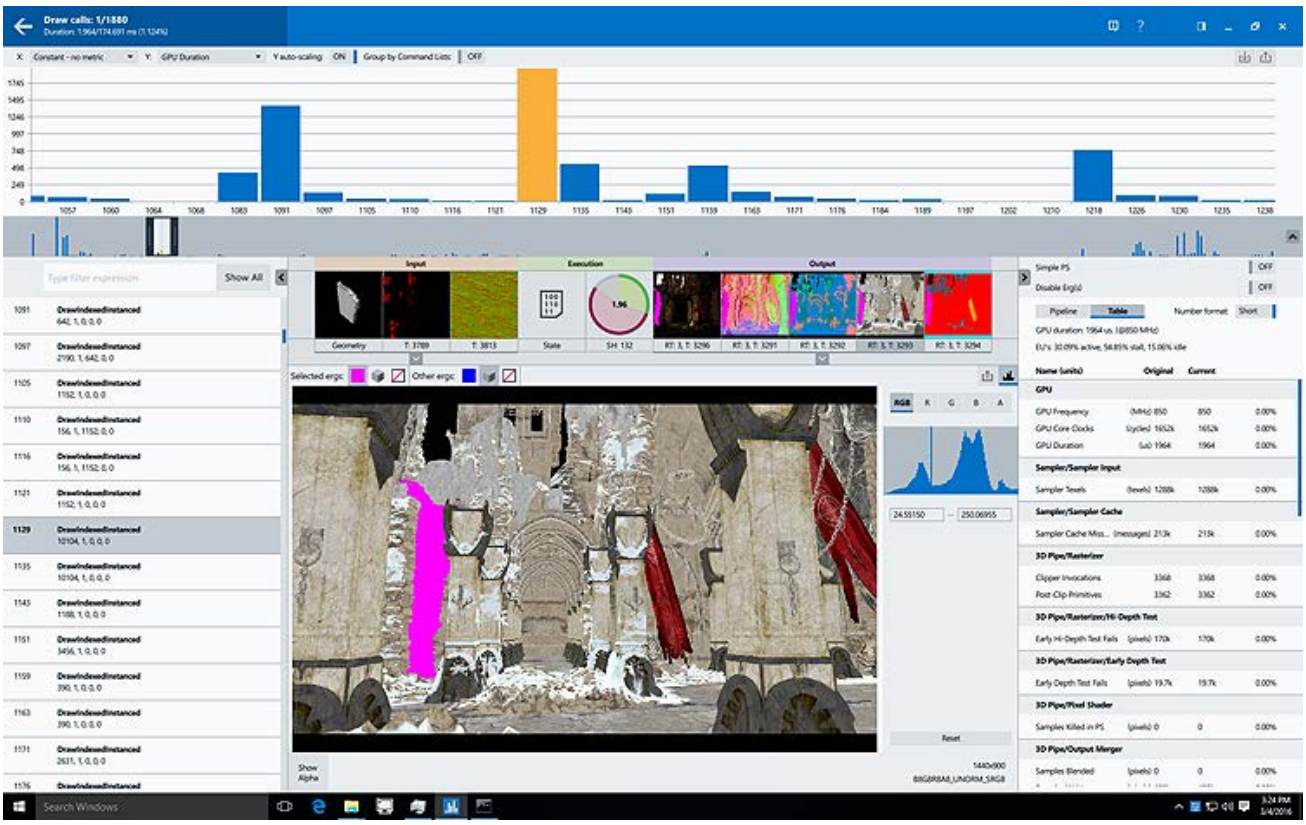


図 6. インテル® グラフィックス・パフォーマンス・アナライザーを使用してフレームレベルでボトルネックを特定および軽減

VR アプリケーション・チューニングのレシピ

VR アプリケーションを最適化する場合、[Key performance metrics (主要パフォーマンス・メトリック)] セクションの**主要メトリック**を使用して初期パフォーマンスを測定し、各レベルで変更の影響を調べます。

主要パフォーマンス・メトリック

フレームレート

1 秒あたりにレンダリングされるフレーム数として定義され、一般に fps (frames per second、フレーム毎秒) と呼ばれます。fps は高いほうが優れています。

Late Stage Reprojection (LSR)

LSR は Windows Mixed Reality (WMR) アプリケーションにのみ適用されます。ヘッドセットに渡す前に画像を再投影できるかどうか判断するには、LSR をキャプチャーします。再投影を行うと、VR 酔いが軽減されます。LSR は高いほうが優れています。

フレーム時間

フレームをレンダリングする合計時間。ターゲット・フレームレートが 90 fps の場合は 11.1 ミリ秒、60 fps の場合は 16.6 ミリ秒で完了する必要があります。フレーム時間は低いほうが優れています。

特定の fps で必要な時間を決定するには、次の計算を行います。
1000 / (ターゲット fps) = X ミリ秒
例: 1000 / 90 fps = 11.1 ミリ秒

CPU と GPU 使用率

指定された時間にプロセシング・ユニットで処理された時間の合計。

成功の定義

VR アプリケーションの主要パフォーマンス・メトリックを正しく理解し、ターゲット・ハードウェアの達成可能な最大パフォーマンスを認識して、パフォーマンスの上限を判断します。さまざまなメトリックについてこの上限を計算し、状況に応じて実施に測定したパフォーマンスを評価します。例えば、ターゲット・プラットフォームで 98% 以上のフレームレートと Late Stage Reprojection (LSR) 値を達成すれば、アプリケーションは適切に動作します。達成可能な最大リフレッシュ・レートが 90Hz の場合は、少なくとも 88.2 fps のフレームレートと LSR 平均を達成するようにします。Late Stage Reprojection は Microsoft* による造語で、Windows* Mixed Reality アプリケーションに使用されます。しかし、Oculus* および Vive* アプリケーションは、独自の再投影用語とソリューションを使用しています。この記事では、LSR を使用して WMR アプリケーションの理想的なパフォーマンスを説明します。

パフォーマンスのレベル

多くの主要 VR プラットフォームは異なるレベルのパフォーマンスを提供します (表 3 を参照)。

例えば、Microsoft* は 2 つの Mixed Reality PC (Windows* Mixed Reality Ultra PC (WMR Ultra PC) および Windows*Mixed Reality PC (WMR メインストリーム PC) を定義しています。主な相違点は、最小ハードウェア要件 (英語) と達成可能なヘッドセットの最大フレームレートです。WMR メインストリーム PC は 60Hz をサポートし、WMR Ultra PC は 90Hz をサポートします。

ほかの最小ハードウェア要件については、Oculus* Rift サポートページおよび HTC Vive* の推奨仕様のリストを参照してください。

表 3. 達成可能な最大 fps パフォーマンス

	WMR メインストリーム PC	WMR Ultra PC	Oculus* Rift*	HTC Vive* & Vive* Pro
最大 fps	60	90	90	90

VR チューニング・フロー

主要パフォーマンス・メトリックとアプリケーションが意図したとおりに動作する一般的な指標を確認したら、最小仕様に基づいてテストを開始します。目標は、各ターゲット・プラットフォームで優れたユーザー・エクスペリエンスを提供することです。

ベンチマークとチューニングで、VR アプリケーションに修正すべきパフォーマンス問題があるかどうか判断します。図 7 は、解析とチューニング・プロセスの一般的なフローを示しています。使用するツールは図の上部に示されています。

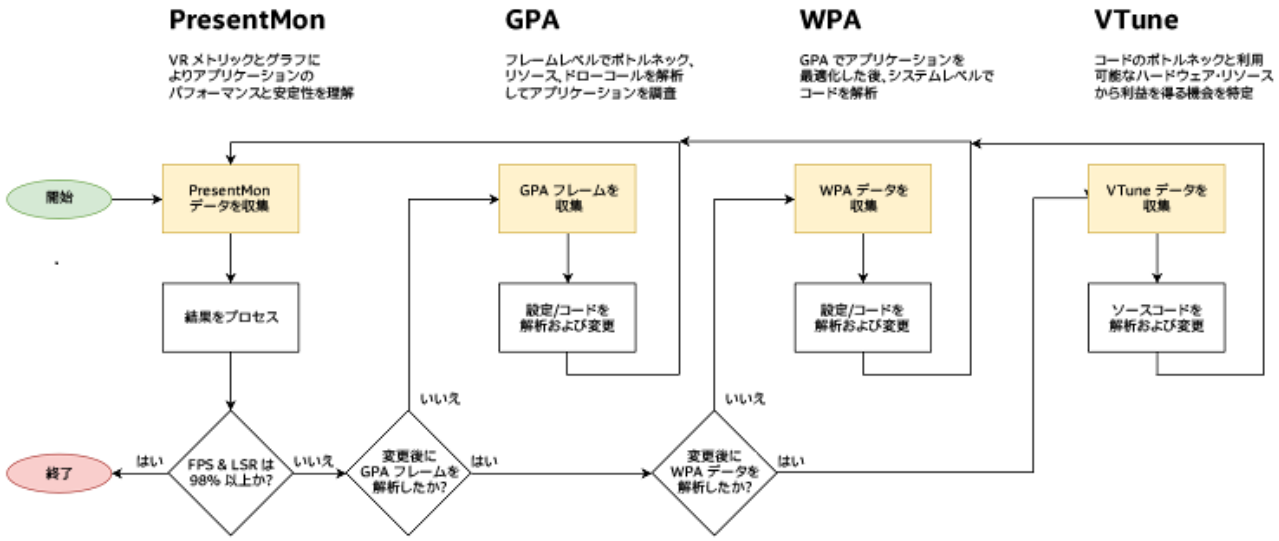


図 7. VR アプリケーション解析とチューニングのフローチャート

はじめに

PresentMon を使用してアプリケーションの動作を明らかにした後、さらに最適化が必要かどうか判断します。PresentMon データを使用してグラフを生成し (図 8)、テスト全体のアプリケーションの動作を理解します。このグラフから、バッファリング問題とスタッタリング問題、およびパフォーマンスの変動を瞬時に識別できます。これらのメトリックの平均をとると、特定の問題およびそれらがユーザー・エクスペリエンスに与える影響が明らかにならないことがあります。

平均フレームレートは 55 fps を上回っていますが (図 8)、頻繁な変動があることに注意してください。大きな変動は、その期間に多くのフレームがドロップしたことを表します。フレームがドロップすると、スタッタリング問題が発生し、ユーザー・エクスペリエンスが低下します。

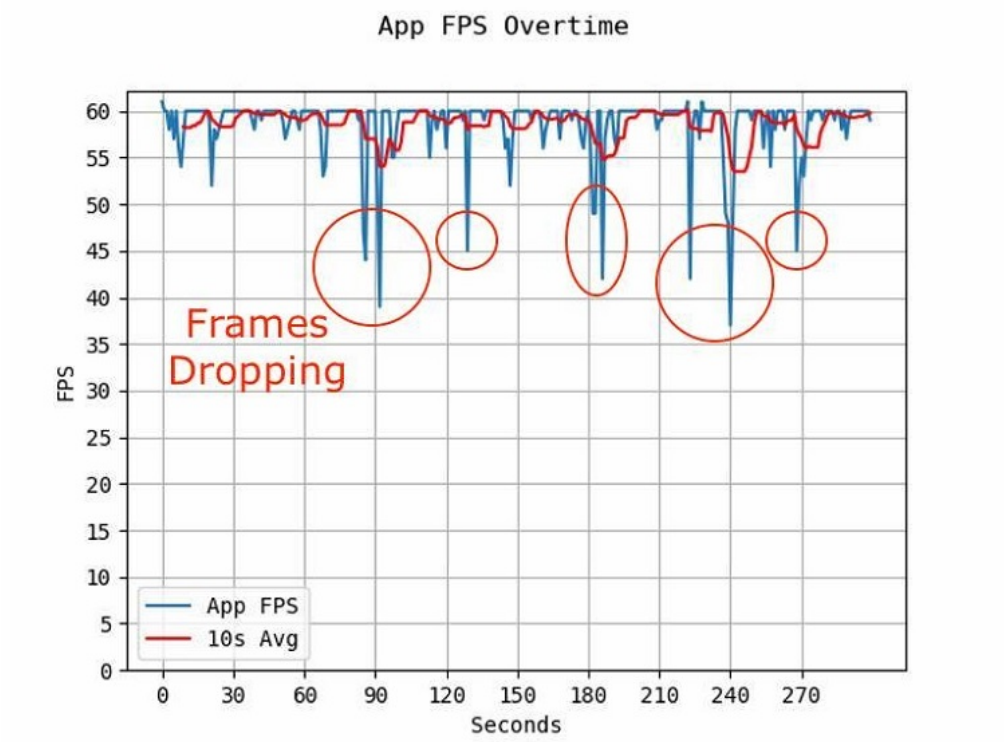


図 8. 収集された PresentMon データに基づいて生成されたグラフ

操作手順:

1. **PresentMon** (英語) の最新バージョンをダウンロードします。
2. コマンドラインを使用して**少なくとも 90 秒のクリップ**を収集し、アプリケーションからサンプルを収集します。
3. キャプチャー・コマンド例: **sampleapp.exe: .\PresentMon64-1.3.0.exe -timed 90 -process_name sampleapp.exe -verbose**
    1. Windows Mixed Reality (MR) アプリケーションの場合は、**-include_mixed_reality** オプションを使用して追加のメトリックを収集します。
4. データを処理します。ここでは Excel* を使用してアプリケーション・パフォーマンスのグラフを表示していますが、ほかの表計算アプリケーションを使用してもかまいません。
    1. PresentMon により生成された **.csv** ファイルを開きます。
    2. [Application (アプリケーション)] 列 (列 A) で、列が解析している VR アプリケーションで**フィルター**されていることを確認します。
    3. **新しい列**を作成し、名前を fps に設定して、計算: =1000/[MsBetweenPresents] をすべての行に適用します。
5. データをグラフにします。

1. Excel* で、**新しく作成した fps 列**をハイライトして、**[挿入] - [2D 折れ線]** を選択します。ウィンドウの右下に表示される平均に注意してください。

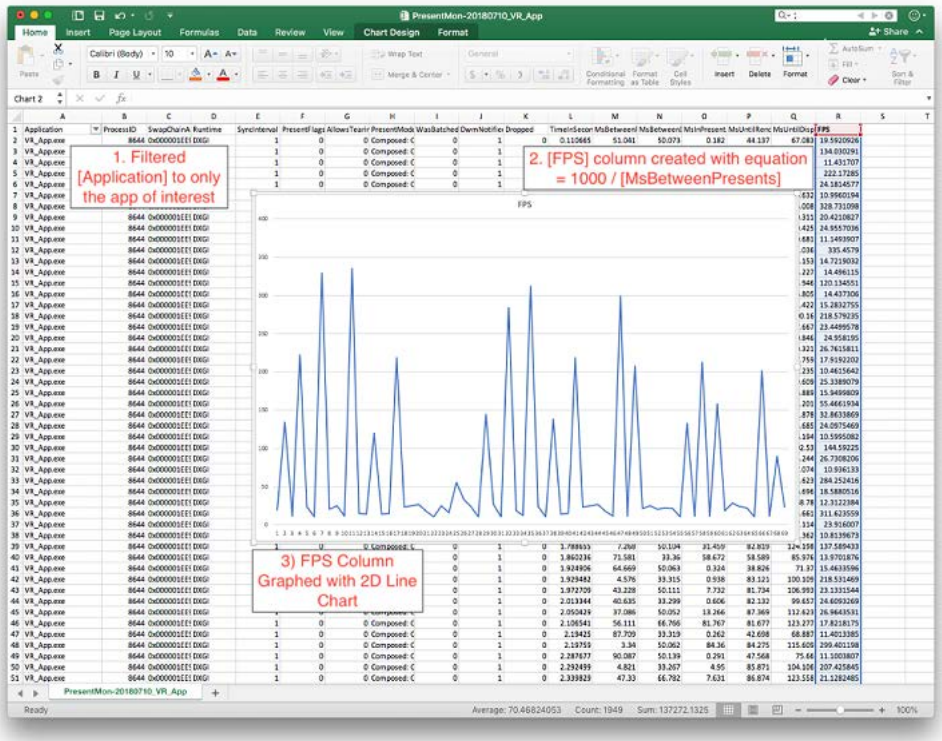


図 9. Excel* で生成したグラフ

PresentMon データの解析

アプリケーションの fps グラフ (図 10) を見ると、パフォーマンス問題を簡単に判断できます。アプリケーションが理想的な方法で実行されている場合、fps 平均は達成可能な最大 fps の 98% 以上になり、fps 標準偏差は低くなります。

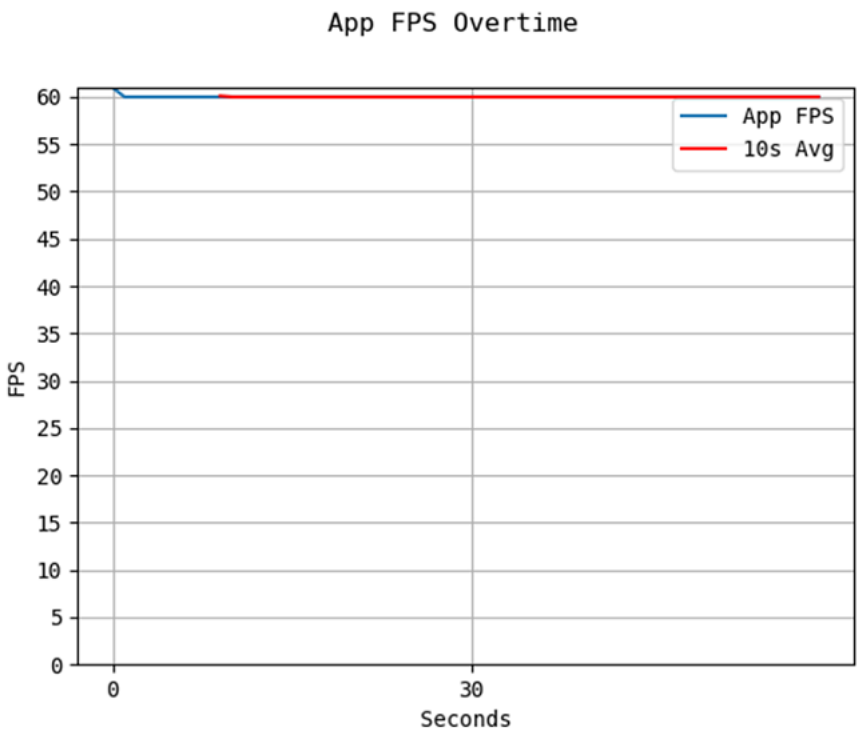


図 10. 達成可能な最大 fps が 60 fps で 10 秒平均が安定している理想的なアプリケーションの例

平均 fps が 50 台後半でも、断続的にフレームがドロップするケースを考えます。図 11a および 11b は次のことを示しています。

(a) 動的な画質設定の変更により 90 秒ごとにフレームがドロップしています。

(b) フレームはドロップした後、数秒後に安定しています。バッファリング問題の可能性があります。

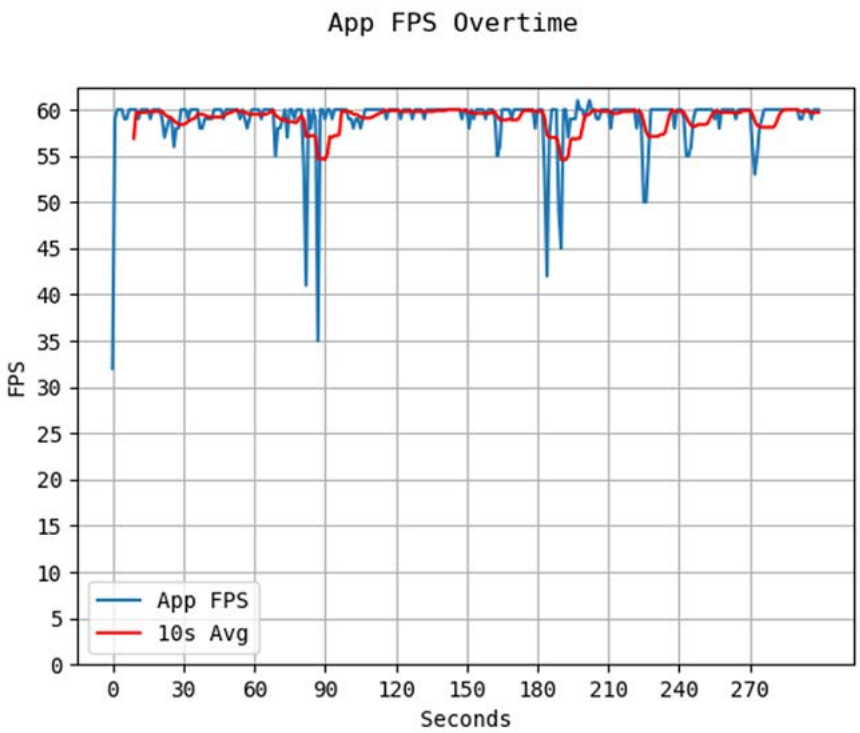


図 11a. 90 秒ごとにフレームがドロップしている

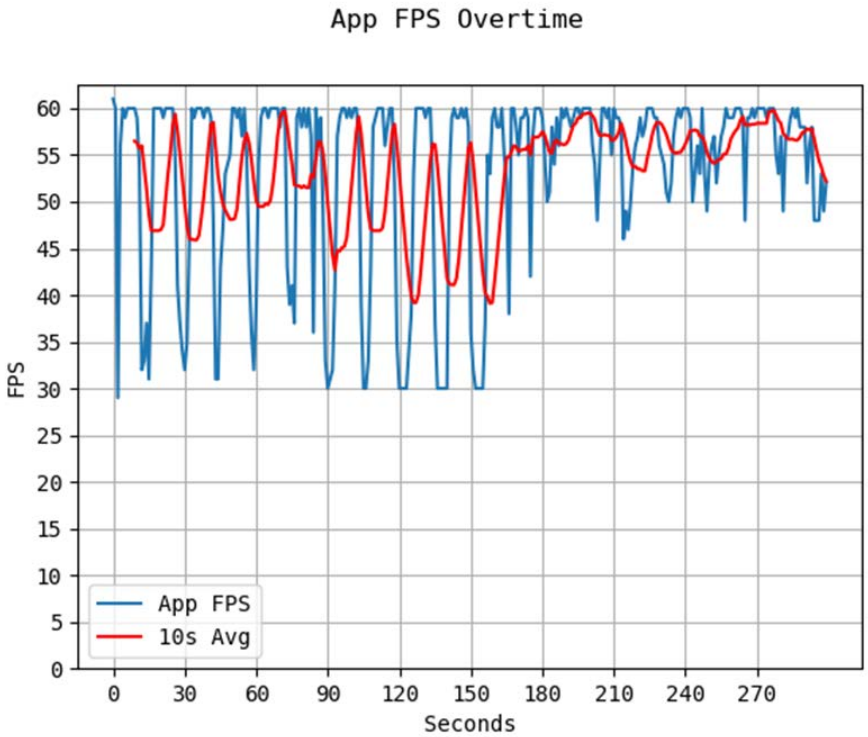


図 11b. 時間が経過するとスムーズになるバッファリング問題

アプリケーションが常に fps の目標を達成できない場合、インテル® GPA を使用して調査します。

インテル® グラフィックス・パフォーマンス・アナライザー (インテル® GPA)

インテル® GPA を使用してフレームをキャプチャーし、VR アプリケーションの解析を続けます。最も大きなワークアイテム (erg) を識別します。次のことが明らかになります。

- 不要な背景レンダリング
- 非効率にバッチ処理されたドローコール
- パフォーマンスが低下する可能性があるほかの問題

VR アプリケーションでは、erg が 2 回現れることがあります。これは、シングルパス・ステレオ・レンダリング (erg が 1 回のみ現れる) を使用しない限り、左右の目でそれぞれ 1 回レンダリングされるためです。シングルパス・ステレオ・レンダリングを使用して CPU と GPU 使用率を下げることを推奨します。詳細は、Unity* のユーザーマニュアルを参照してください。

図 12 は、メディアプレーヤーのメニューがビデオの後ろでレンダリングされている (ユーザーが見ない) 典型的な状況を示しています。インテル® GPA は、ユーザーが最終的な出力でプレーヤーがレンダリングされているのを見ないことを確認して、ドローコールやレンダリング・ターゲットを表示できます。

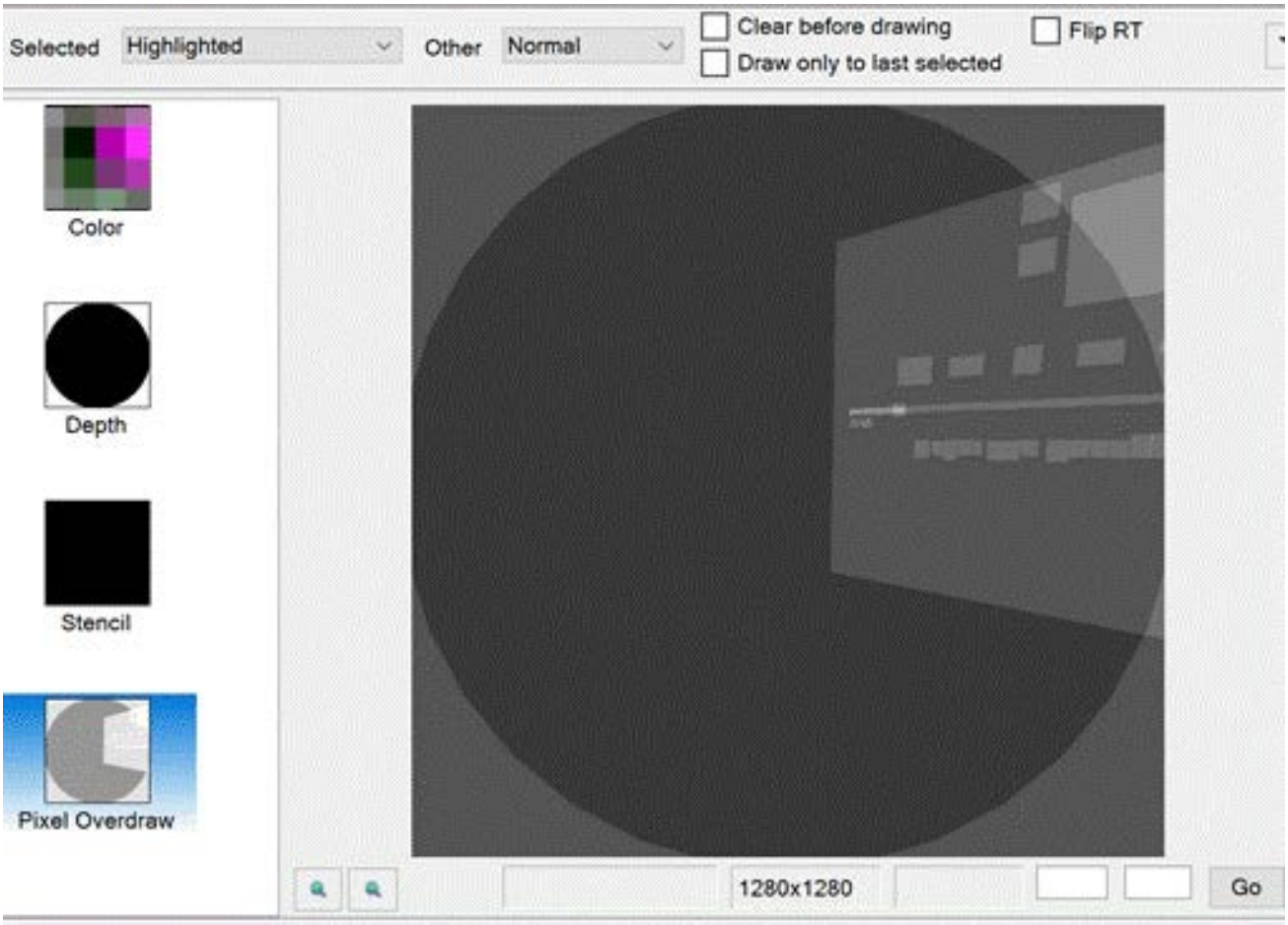


図 12. ビデオの後ろでレンダリングされているメディアプレーヤーのメニュー

図 13a、13b、13c は、インテル® GPA がターゲット・ハードウェア向けに非常に高く設定された静的 RenderScale を検出する方法を示しています。このインスタンスでは、RenderScale が 1.3 に設定された後、ターゲットサイズにダウンスケールされていました。RenderScale を 1.0 に戻すことで、大幅にパフォーマンスが向上しました。

ローエンドのハードウェアでは、VR でより重要な、安定した fps を維持したままで、RenderScale を 0.7 またはわずかに高い値に動的に下げると、画質の低下を最小限に抑えることができます (ユースケースに依存します)。VR 酔いの対処方法は、このビデオ (英語) を参照してください。

図 13a. レンダーターゲット間で解像度が 1664x1664 から 1280x1280 に変更されている (RenderScale が 1 よりも大きな値に設定されている)

PresentMon データは、RenderScale を 1.0 に戻した後にパフォーマンスが向上したことを示しています。

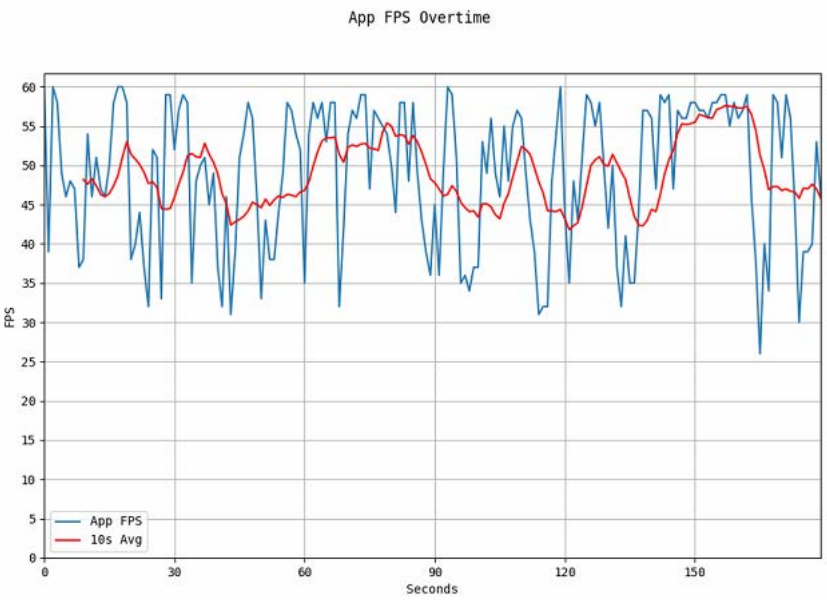


図 13b. RenderScale 1.3 の平均フレームレートは 48.9 fps

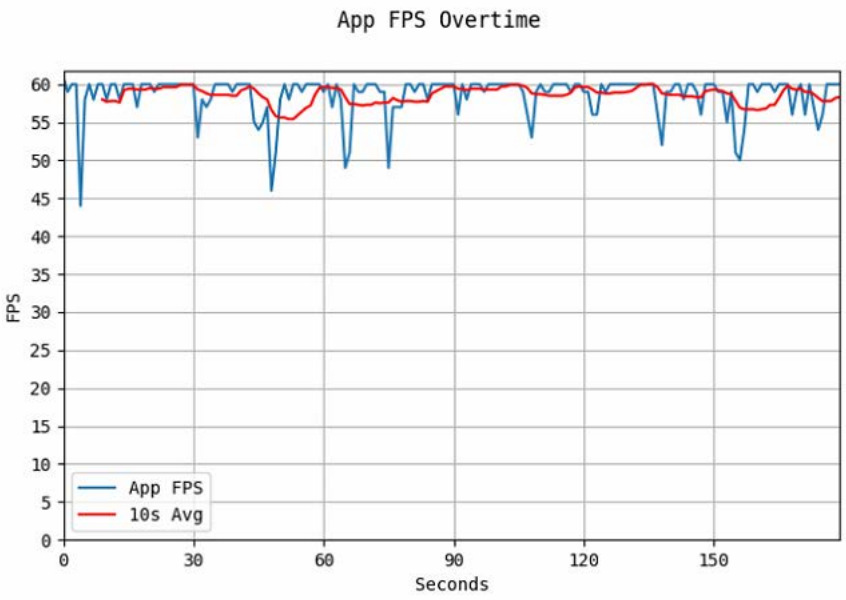


図 13c. RenderScale を 1.0 に調整した後の平均フレームレートは 58.6 fps

インテル® GPA を使用して、最も大きな erg でドローコールが何を行っているか調べます。次に、ボトルネック (例えば、大きなテクスチャーによりストールしているシェーダー) を探します。最後に、erg を調べて変更によりパフォーマンスが向上したかどうかを確認します。この記事のフレーム内のパフォーマンス・ボトルネックを正確に特定する方法の詳細は、インテル® GPA 入門 (英語) を参照してください。

操作手順:

1. インテル® グラフィックス・パフォーマンス・アナライザーを**ダウンロード** (英語) します。
2. **インテル® GPA から**フレームを解析するアプリケーションを起動します。
3. アプリケーションが起動したら、パフォーマンス問題が想定されるシーンで **(Ctrl+C)** または **(Cmd+C)** キーを押してフレームをキャプチャーします。
4. **Graphics Frame Analyzer** でキャプチャーしたフレームを表示してフレームレベルで調査します。

インテル® GPA を使用したアプリケーション・レベルの最適化が要件を満たさない場合、Windows* パフォーマンス レコーダーを使用してシステムレベルで調査します。

Windows* パフォーマンス レコーダー(WPR)

WPR でテスト中のデータを記録して WPA で解析します。WPA は、CPU と GPU の使用状況とイベントのグラフィカルな表現を表示します。GPU ビューと Xperf に加えて、WPA と WPR には Windows* ADK の一部として利用可能な Windows* パフォーマンス・ツールキットが付属しています。

WPR GUI の [Resource Analysis (リソース解析)] で [GPU activity (GPU アクティビティー)] オプションを選択していることを確認します。このオプションはデフォルトでは選択されません。

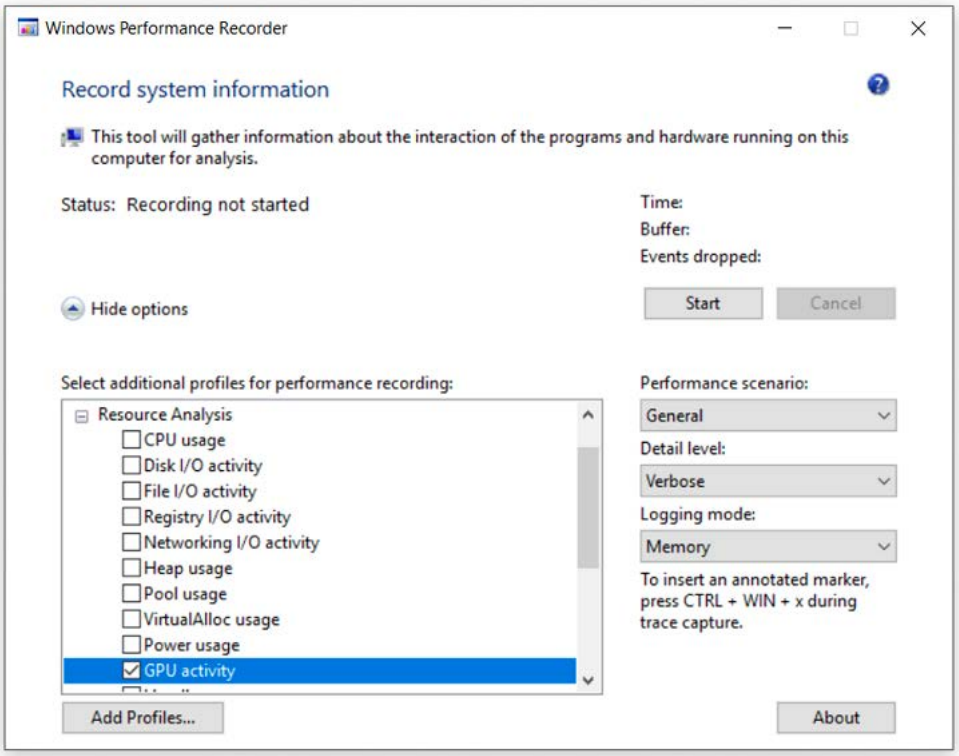


図 14 は、(黄色でハイライトされた) アプリケーション WWAHost.exe を実行したときにキャプチャーされた WPA トレースを示しています。[CPU Usage (sampled) (CPU 使用率 (サンプリング))] フィールドの [% Weight (% ウェイト)] 列から、このアプリケーションのウェイトが 10.56% であることがわかります。GPU 使用率は、左の [Graph Explorer (グラフ・エクスプローラー)] の [Video (ビデオ)] メニューの下に表示されます。

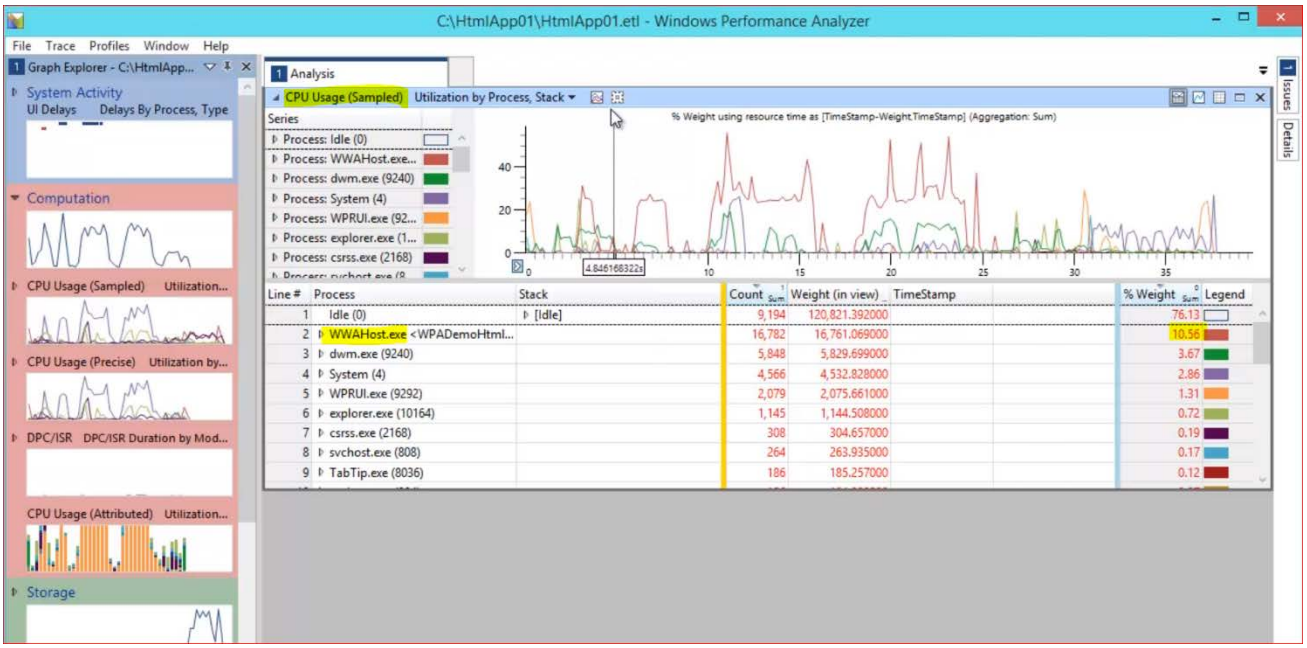


図 14. WPA で CPU と GPU の使用状況を解析

ソリューション・シナリオ

我々は、パフォーマンス・エンジニアとして、ほかの開発者がアプリケーションを適切に最適化できるように支援します。VR アプリケーションの最適化に最適ないくつかの手法を次にリストします。

GPU 使用状況のチューニング

GPA のボトルネック

潜在的な問題点を探します。ポータルおよび不要な要素をバックグラウンドでレンダリングしたり、使用されていないテクスチャーをロードすると、パフォーマンスは低下します。Graphics Frame Analyzer を使用して、フェッチされるテクスチャー、ドローコールのバッチ処理、不要なクリアコールを探し、最も大きな影響を与えているドローコールと、不可欠な要素がシーンでレンダリングされているかどうか判断します。

下記の図のように、Graphics Frame Analyzer で X と Y メトリックの変更と GPU 時間の両方に注目して、レンダリング時間が最も長いドローコールを特定します。パイプライン・ボトルネックをピンポイントで特定するテストを行います。

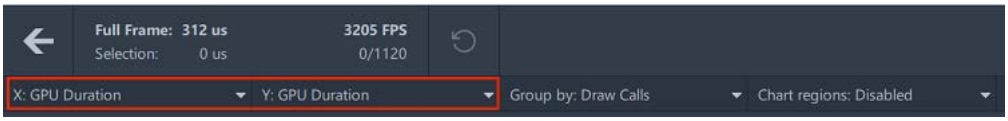


図 15. GPU 時間を表示するように X と Y ドロップダウンを変更して時間が最も長い erg を特定する

WPA

問題

図 16 は、アプリケーションの GPU 使用率がほぼ 100% になっていることを示しています。これは、アプリケーションが GPU 依存であることを意味します。この場合、CPU 使用率を確認します。CPU 使用率に余裕がある場合は、GPU ワークロードを CPU にオフロードします。

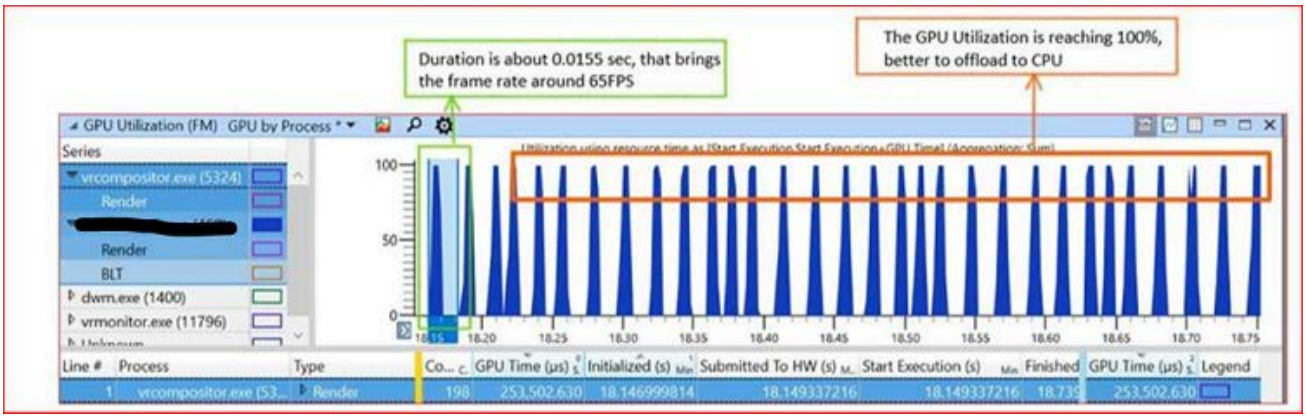


図 16. GPU 使用率がほぼ 100% の場合 (オレンジでハイライト) アプリケーションは GPU 依存

起動時間の短縮

多くの要因が VR アプリケーションの起動時間に影響します。多くの大きなアセットファイルは大きなパフォーマンスの低下をもたらします。高性能のハードウェアでも、VR アプリケーションの起動には 30 ~ 60 秒またはそれ以上かかることがあります。起動時間問題のシナリオ、根本的な原因、回避方法を調べることにしましょう。

問題のシナリオ: シリアル化されたネットワーク操作

この問題は VR アプリケーションの任意の場所で発生しますが、最も厄介なのは起動中に発生した場合です。WPR で収集し、WPA で表示した図 17 の例を考えます。

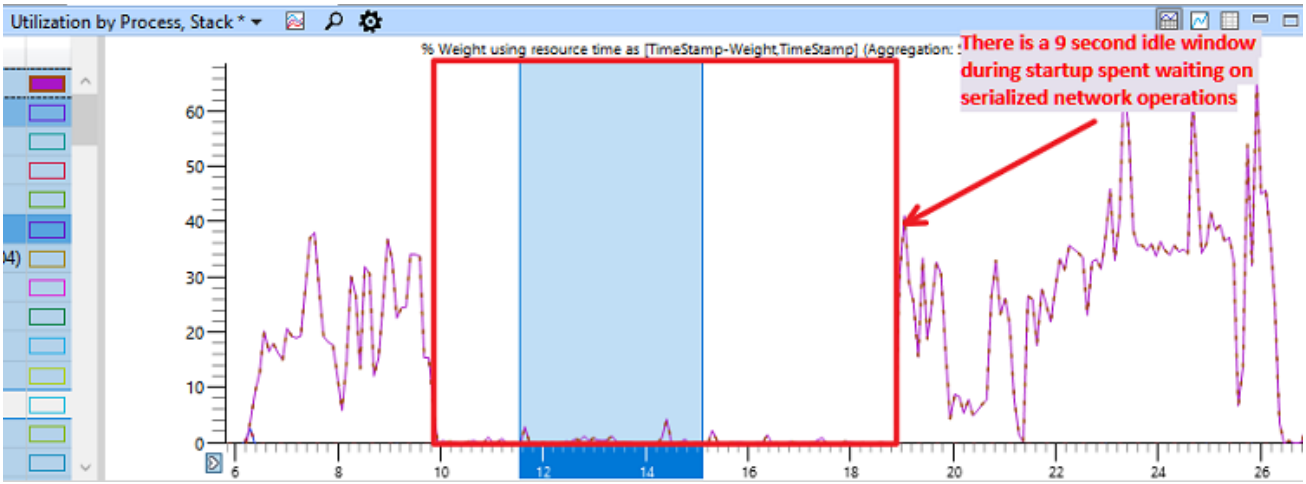


図 17. シリアル化されたネットワーク操作は VR アプリケーションの起動時間に長いレイテンシーを追加する

この例では、起動時に 9 秒間 CPU と GPU がアイドルで待機しています。この赤のフラグは、ボトルネックがプロセッサー以外の部分であることを意味します。開発者は、特に不足しているものがあるとは考えず、起動時間を短縮する方法はないと仮定していました。

これは重要なルールを示しています。適切にチューニングされたアプリケーションは、ボトルネックになる前に目的のユーザー・エクスペリエンスが達成される唯一の例外を除いて、常に CPU または GPU により制限されるべきです。

WPA で動作を識別することで、9 秒間の根本的な原因は簡単に見つけることができました。ソースコードの解析により、問題はアップデート操作であることが判明しました。独自のスレッドで実行するはずのアップデートが UI スレッドで実行されていました。10 分かけて、3 行のコードを修正しました。アップデート操作をオフロードするスレッドをスポーンして、UI スレッドをブロックしないようにしました。

図 18 は、変更後のスタートアップ・プロファイルの結果です。

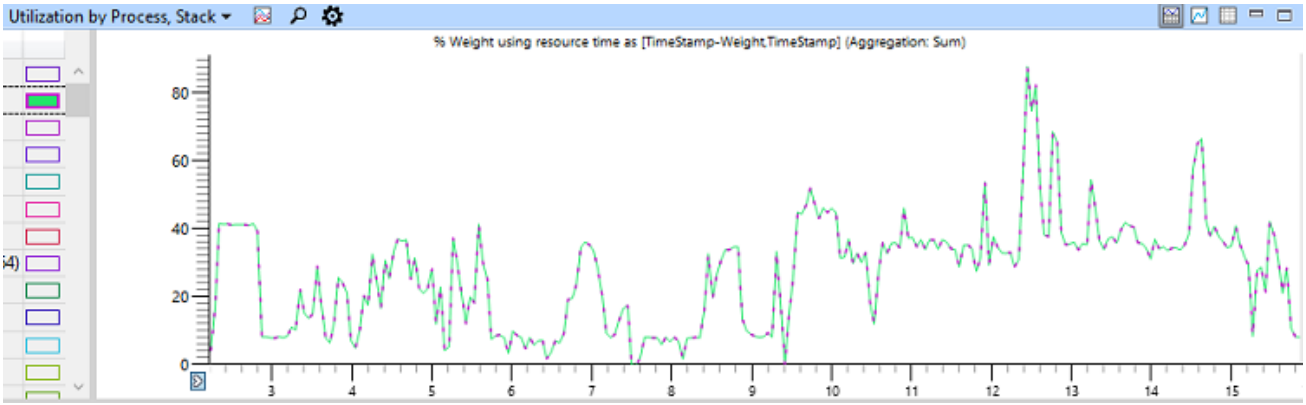


図 18. 起動時のボトルネックの原因を取り除くことにより起動時間を 11 秒短縮

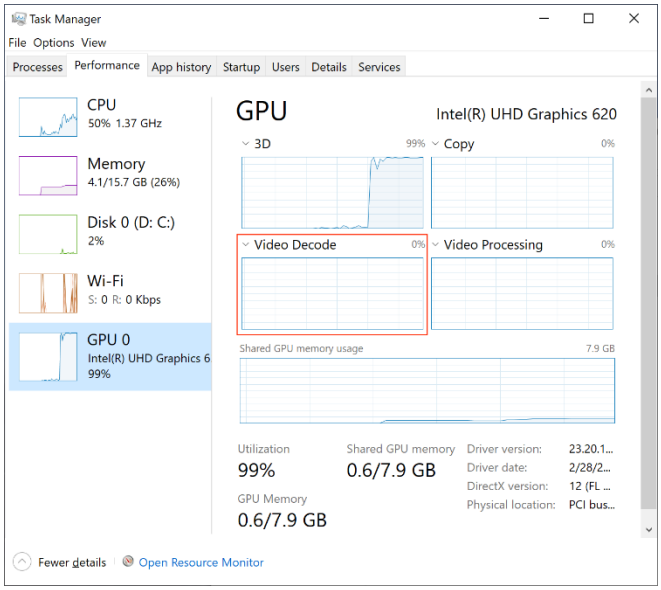
CPU 使用率のギャップがなくなったことに注目してください。起動時間は 26 秒以上から 16 秒未満に短縮されました。このパフォーマンスの向上は、アップデート操作を非同期に実行することにより得られたものです。

必要な場合を除いて、起動中にメインまたは UI スレッドをブロックしないように、常にネットワーク操作を並列化して別のスレッドで実行してください。最良の最適化の可能性は、ありふれた光景の中に隠れていることもあります。

ビデオ再生の CPU 使用率の軽減

問題: アプリケーションのビデオ再生の CPU 使用率が高い。

ソリューション: タスク・マネージャーの [パフォーマンス] タブで、ハードウェア・デコードを使用していることを確認します (図 19 を参照)。



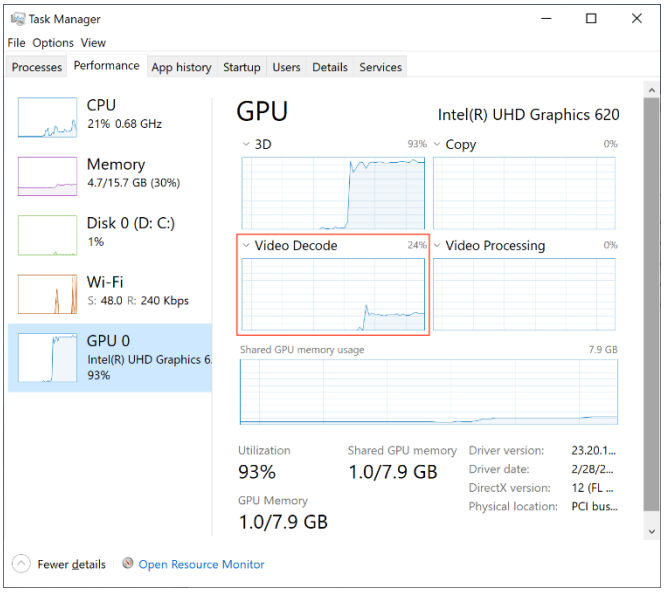


図 19. ハードウェア・デコードなし (上) とハードウェア・デコードあり (下)

ハードウェア・アクセラレーションがエディターで利用できない場合、設定のオプションを確認します。Unity* では、H.264 ソースを使用することが推奨されています。Unity* に H.264 ビデオのハードウェア・デコードが存在しない場合、このサンプル (英語) を使用してアプリケーションに統合します。

まとめ

この記事は、VR アプリケーションを最適化するための基本的な手法、ツール、テクニックを VR アプリケーション開発者に紹介する出発点です。すべてのアプリケーションは異なります。この記事に含まれるレシピとソリューションは、すべてのケースおよびアプリケーションに適しているとは限りません。この記事で示した状況の種類と方向性に応じて、上記のセクションで説明した異なるツールと手法を使用してアプリケーションを解析してください。この記事で説明したすべてのツールのリンクを、この後にリストしました。特定の最適化トピックの詳細は、関連情報を参照してください。

関連資料 (英語)

- ソフトウェア・デベロッパーズ・マニュアル
- ソフトウェア最適化リファレンス・マニュアル
- VR コンテンツ・デベロッパー・ガイド

ツール

- Windows* アセスメント & デプロイメント・キット (Windows* ADK)
- インテル® グラフィックス・パフォーマンス・アナライザー (インテル® GPA)
- PresentMon (英語)

関連情報

- Oculus* デバッグツール (英語)
- FCAT キャプチャーと解析ツール (英語)
- Windows Mixed Reality で SteamVR を使用する (英語)
- Unity* グラフィックス・パフォーマンスの最適化ガイド
- Unity* テクスチャー・インポーター・マニュアル
- Unity* ポストプロセシング・ユーザー LUT マニュアル
- Unity* シングルパス・ステレオ・レンダリング

ベスト・プラクティス・チェックリスト

品質とレンダリング設定: WMR メインストリーム PC

HDR	無効。RGBA 8 または 10
MSAA	無効
メディア入力	2K
異方性フィルタリング	無効
垂直同期	無効
シャドウカスケード	無効
テクスチャー	低または中
シングルパス・ステレオ・レンダリング	有効
レンダースケール	0.7 - 1.0

Unity* アプリケーション・チューニングのヒント

バーテックス・シェーダーが pow、exp、log、cos、sin、tan などの数学関数を含む場合。

ユーザー LUT を使用したポストプロセシングが可能な場合、複雑な数学計算の代わりにルックアップ・テクスチャーを使用することを検討します。

サンプラー・ボトルネックがある場合、データの取得が遅いためにテクスチャー・サンプラーによる EU スタベーションが発生します。

- 低い解像度や RGBA8 のような色精度を使用して、テクスチャーのサイズを小さくすることを検討します。
- 異なるフィルタリング・アルゴリズムを使用することを検討します。例えば、異方性フィルタリングはバイリニア・フィルタリングのような単純なアルゴリズムよりもコストがかかります。

    - これが問題かどうか確認するには、2x2 のテクスチャーをテストして、変更がフレームの時間に影響するかどうか確認します。
- 非常に多くのドローコールにより CPU 側で著しいオーバーヘッドが発生している場合で、アプリケーションが Unity* を使用している場合は、動かないゲーム・オブジェクトに静的バッチ処理を使用することを検討します。
    - 動かないゲーム・オブジェクトが Static (静的) になっていることを確認します。
- 広範なシステム仕様をターゲットにする場合は、RenderScale 設定を動的に変更して fps を下げることを検討します。0.7 または 0.8 に下げても顕著な違いがない場合、低スペックのシステムのオプションとして利用できます。画像を小さくすることはフレームのドロップを回避する 1 つの方法です。
- シングルパス・ステレオ・レンダリングを使用して CPU 処理時間を短縮します。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。