第 2 章 Visual Profiler

NVIDIA Visual Profiler を使用すると、アプリケーションの性能を視覚化して最適化できる
Visual Profiler は、CPU と GPU の両方でのアプリケーションのアクティビティのタイムラインを表示すため、性能改善の機会を特定できる
さらに、Visual Profiler はアプリケーションを分析して潜在的な性能のボトルネックを検出し、それらのボトルネックを解消または削減するための対策を講じる
Visual Profiler は、スタンドアロンアプリケーションとしても、Nsight Eclipse Edition の一部としても利用できる
スタンドアロンバージョンの Visual Profilernvvp は、サポートされているすべての OS の CUDA ツールキットに含まれる
Nsight Eclipse Edition では、Visual Profiler はプロファイルパースペクティブにあり、アプリケーションがプロファイルモードで実行されるとアクティブになる
Visual Profile を実行する前に、ローカルシステムに Java ランタイム環境(JRE)1.8 が必要なことに注意
CUDA ツールキット v10.1 以降、Oracle のアップグレードライセンスの変更により、JRE は CUDA ツールキットに含まれない
Visual Profiler を使用するには、JRE 1.8 をインストールする必要がある
JRE のインストールを参照のこと www.java.com

OpenSUSE15 または SLES15 で Visual Profiler を実行するには:
以下コマンドラインオプションを含めて入力
nvvp -vm /usr/lib64/jvm/jre-1.8.0/bin/java
-vm オプションは、JRE が CUDA ツールキットパッケージに含まれず、JRE 1.8 がパスにない場合に必要

Ubuntu 18.04 または Ubuntu 18.10 で Visual Profiler を実行するには:
以下コマンドラインオプションを含めて入力
nvvp -vm /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
-vm オプションは、JRE が CUDA ツールキットパッケージに含まれず、JRE 1.8 がパスにない場合に必要

Ubuntu 18.10 では、Visual Profiler 実行時に「no swt-pi-gtk in java.library.path」エラーが表示される場合は、GTK2 をインストールする必要がある
以下のコマンドを入力して、GTK2 をインストールする
apt-get install libgtk2.0-0

Fedora 29 で Visual Profiler を実行するには
以下コマンドラインオプションを含めて入力

nvvp -vm/usr/bin/java
-vm オプションは、JRE が CUDA ツールキットパッケージに含まれず、JRE 1.8 がパスにない場合に必要

MacOSX で Visual Profiler を実行するには:
以下コマンドラインオプションを含めて入力
nvvp -vm/usr/bin/java
-vm オプションは、JRE が CUDA ツールキットパッケージに含まれず、JRE 1.8 がパスにない場合に必要

Windows で Visual Profiler を実行するには:
以下コマンドラインオプションを含めて入力
nvvp -vm 'C:¥ Program Files¥Java¥jdk1.8.0_77¥jre¥bin¥java'
-vm オプションは、JRE が CUDA ツールキットパッケージに含まれず、JRE 1.8 がパスにない場合に必要

2.1 入門
この章では、プロファイリングの開始手順を説明する
2.1.1 プロファイリングのためのアプリケーションの変更
Visual Profiler では、アプリケーションを変更する必要はない
ただし、簡単な変更と追加を行うことで、その使いやすさと効果を大幅に向上させる
「プロファイリングのためのアプリケーションの準備」では、プロファイリング作業に集中し、プロファイリングを大幅に向上させる追加の注釈をアプリケーションに追加する方法について説明する

2.1.2 セッションを作成する
Visual Profiler を使用してアプリケーションのプロファイルを作成する最初のステップは、新しいプロファイリングセッションの作成である
セッションには、アプリケーションに関連付けられた設定、データ、結果が含まれる
セッションの章では、セッションの操作について詳しく説明する
「ようこそ」ページで「アプリケーションのプロファイル」リンクを選択するか、「ファイル」メニューから「新規セッション」を選択して、新しいセッションを作成する
[新しいセッションの作成]ダイアログで、アプリケーションの実行可能ファイルを入力する
オプションで、作業ディレクトリ、引数、マルチプロセスプロファイリングオプション、および環境を指定する

マルチプロセスのオプションは次の通り
Profile child processes - 指定したアプリケーションによって起動されたすべてのプロセスをプロ

ファイルする
Profile all processes - nvprof を起動した同じユーザーが同じシステムで起動したすべての CUDA プロセスをプロファイルする
このモードでは、Visual Profiler が nvprof を起動し、ユーザーはアプリケーションを Visual Profiler の外部の別の端末で実行する必要がある
ユーザーは、Visual Profiler の進行ダイアログの[キャンセル]ボタンを押してこのモードを終了し、プロファイルデータをロードできる
Profile current process only - 指定したアプリケーションのみをプロファイルする

[Next]を押して、追加のプロファイリングオプションを選択する

CUDA オプション:
Start execution with profiling enabled
選択された場合、プロファイルデータがアプリケーションの実行開始から収集される
選択されない場合、アプリケーションで cudaProfilerStart()が呼び出されるまでプロファイルデータは収集されない
cudaProfilerStart()の詳細は集中プロファイリングを参照のこと

Enable concurrent kernel profiling
同時カーネルプロファイリングを有効にする
CUDA ストリームを使用して、同時に実行できるカーネルを起動するアプリケーションでは、このオプションを選択する必要がある
アプリケーションが単一のストリームのみを使用する(したがって、カーネルを同時に実行できない)場合、このオプションを選択解除すると、プロファイリングのオーバーヘッドが減少する可能性がある

Enable CUDA API tracing in the timeline
タイムラインで CUDA API トレースを有効にする
選択すると、CUDA ドライバとランタイム API 呼び出しのトレースが収集され、タイムラインに表示される

Enable power, clock, and thermal profiling
電力、クロック、および熱プロファイリングを有効にする
選択すると、GPU の電力、クロック、および熱条件がサンプリングされ、タイムラインに表示される
このデータの収集は、すべての GPU でサポートされていない
詳細は Timeline View のデバイスタイムラインの説明を参照のこと

Enable unified memory profiling
ユニファイドメモリプロファイリングを有効にする
ユニファイドメモリをサポートする GPU に対して選択した場合、各 GPU との間のユニファイドメモリ関連のメモリトラフィックがシステムで収集され、タイムラインに表示される

Replay application to collect events and metrics
アプリケーションを再生してイベントとメトリックを収集する
選択すると、すべてのイベント/メトリックを収集するために、各カーネルを再生する代わりにアプリケーション全体が再実行される

Run guided analysis
ガイド付き分析の実行
選択すると、新しいセッションの作成直後にガイド付き分析が実行される
この動作を無効にするには、このオプションをオフにする

CPU(ホスト)オプション:
Profile execution on the CPU
CPU でのプロファイル実行
選択すると、CPU スレッドがサンプリングされ、CPU 性能について収集されたデータが CPU 詳細ビューに表示される

Enable OpenACC profiling
OpenACC プロファイリングを有効にする
選択すると、OpenACC アプリケーションがプロファイルされ、OpenACC アクティビティが記録され、新しい OpenACC タイムラインに表示される
このデータの収集は、Linux および PGI 19.1 以降でのみサポートされる
詳細は Timeline View の OpenACC タイムラインの Timeline View を参照のこと

Enable CPU thread tracing
CPU スレッドトレースを有効にする
有効にすると、選択した CPU スレッド API 呼び出しが記録され、新しいスレッド API タイムラインに表示される
これには現在、Pthread API、mutexes、条件変数が含まれる
性能上の理由から、同時実行に影響する API 呼び出しのみが記録され、このデータの収集は Windows ではサポートされない
詳細は Timeline View のスレッドタイムラインの説明を参照のこと
このオプションは、CUDA を使用した複数の CPU スレッドを持つアプリケーションの依存関係

分析に必要

Timeline オプション:
Load data for time range
時間範囲のデータをロード
選択した場合、ロードするデータの範囲の開始と終了のタイムスタンプを指定できる
このオプションは、大きなデータのサブセットの選択に役立つ

Enable timelines in the session
セッションでタイムラインを有効にする
デフォルトでは、すべてのタイムラインが有効になっている
タイムラインがチェックされない場合、そのタイムラインに関連付けられたデータは読み込まれず、表示されない

オプションをオフにして一部のタイムラインを無効にすると、このタイムラインデータを使用する分析結果は正しくなくなる

Finish を押す

2.1.3 アプリケーションの分析
セッションの作成時に[Don't run guided analysi ガイド付き分析を実行しない]オプションが選択されない場合、Visual Profiler はすぐにアプリケーションを実行して、ガイド付き分析の最初のステージに必要なデータを収集する
2.4.2 分析ビューの章で説明するように、ガイド付き分析システムを使用して、アプリケーションの性能制限動作に関する推奨事項を取得できる

2.1.4 タイムラインを探索する
ガイド付きの分析結果に加えて、アプリケーションの実行時に発生した CPU および GPU アクティビティを示すアプリケーションのタイムラインが表示される
タイムラインで利用可能なプロファイリング情報を探索する方法は、Timeline View とプロパティビューを参照のこと
タイムラインのナビゲートでは、タイムラインをズームおよびスクロールして、アプリケーションの特定の領域に焦点を合わせる方法について説明する

2.1.5 詳細の確認
分析ビューで提供される結果に加えて、分析の一部として収集された特定のメトリックおよびイベント値を確認できる
メトリックとイベントの値が GPU 詳細ビューに表示される

アプリケーションのカーネルがどのように動作しているかを示す特定のメトリックとイベント値を収集できる
GPU 詳細ビューの章で説明されているように、メトリックとイベントを収集する

2.1.6 大きなプロファイルの読み込みを改善する
一部のアプリケーションは多くの小さなカーネルを起動し、数秒のアプリケーション実行であっても、非常に大きな(数百メガバイト以上)出力が発生しやすくなる
Visual Profiler は、開く/インポートするプロファイルのサイズとほぼ同じ量のメモリを必要とする
「最大ヒープサイズ」設定が指定されない場合、Java 仮想マシンはメインメモリの一部を使用することがある
そのため、メインメモリのサイズによっては、Visual Profiler が一部の大きなファイルを読み込めない場合がある
Visual Profiler が大きなプロファイルのロードに失敗した場合は、JVM がメインメモリサイズに応じて使用できる最大ヒープサイズを設定する
ツールキットのインストールディレクトリにある設定ファイル libnvvp/nvvp.ini を変更する
MacOS では、nvvp.ini ファイルは
/Developer/{cuda_install_dir}/libnvvp/nvvp.app/Contents/MacOS/ にある

nvvp.ini 構成ファイルは以下の通り
-startup
plugins/org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.200.v20140603-1326
-data
@user.home/nvvp_workspace
-vm
../jre/bin/java
-vmargs
-Dorg.eclipse.swt.browser.DefaultType=mozilla

たとえば、JVM に 3 ギガバイトのメモリを使用させるには、#vmargs の後に#Xmx3G を使用して新しい行を追加する
-Xmx 設定は、使用可能なシステムメモリと入力サイズに合わせて調整する必要がある
たとえば、システムに 24 GB のシステムメモリがあり、他のメモリを大量に消費するアプリケーションを Visual Profiler と同時に実行する必要がないことがわかっている場合、プロファイラがそのスペースの大部分になる
したがって、たとえば、最大ヒープサイズとして 22GB を選択し、OS、GUI、および実行されて

いる可能性のあるプログラム用に数ギガバイトを残す

nvvp.ini 構成設定を変更する
デフォルトのヒープサイズ(Java が自動的に起動する 1 つ)を、たとえば 2GB に増やする（-Xms）
デフォルトの 32 ビットモードではなく 64 ビットモードで実行するように Java に指示す(64 ビットシステムでのみ機能する)これは、4GB を超えるヒープサイズが必要な場合に必要である（-d64）
Java のパラレルガベージコレクションシステムを有効にするこれにより、特定の入力サイズに必要なメモリスペースを削減し、メモリエラーをより適切にキャッチできる（-XX:+ UseConcMarkSweepGC -XX:+ CMSIncrementalMode）

注:ほとんどのインストールでは、このファイルを変更するために管理者/ルートレベルのアクセスが必要である
上記に従って変更された nvvp.ini ファイルは次のとおり
-data
@user.home/nvvp_workspace
-vm
../jre/bin/java
-d64
-vmargs
-Xms2g
-Xmx22g
-XX:+UseConcMarkSweepGC
-XX:+CMSIncrementalMode
-Dorg.eclipse.swt.browser.DefaultType=Mozilla

JVM 設定の詳細は Java 仮想マシンのマニュアルを参照のこと
これに加えて、タイムラインオプションを使用して、[Load data for time range 時間範囲のデータをロード]および[Enable timelines in the session セッションの作成]章で説明したセッションのタイムラインを有効にして、ロードおよび表示されるデータを制限する

2.2 セッション
セッションには、アプリケーションに関連付けられた設定、データ、およびプロファイリング結果が含まれる
各セッションは個別のファイルに保存される
そのため、セッションファイルを削除、移動、コピー、または共有するだけで、セッションを削除、移動、コピー、または共有できる
慣例により、ファイル拡張子.nvvp は Visual Profiler セッションファイルに使用される

セッションには、Visual Profiler 内から実行およびプロファイルされるアプリケーションに関連付けられた Executable Session（実行可能セッション）と、nvprof によって生成されたデータをインポートすることによって作成される（Import Session）の 2 種類がある

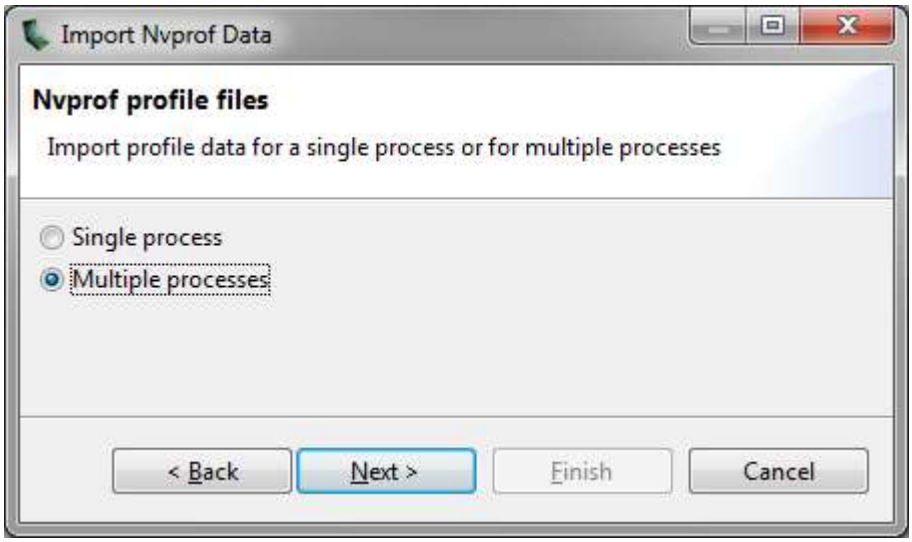
2.2.1Executable Session 実行可能なセッション
「ようこそ」ページで「Profile An Application アプリケーションのプロファイル」リンクを選択するか、「ファイル」メニューから「New Session」を選択することにより、アプリケーションの新しい実行可能セッションを作成する
セッションが作成されたら、設定ビューで説明されているようにセッションの設定を編集する
[ファイル]メニューの[開く]および[保存]オプションを使用して、既存のセッションを開いて保存する
アプリケーションを分析し、メトリックとイベント値を収集するために、Visual Profiler はアプリケーションを複数回実行する
正確なプロファイリング結果を得るには、アプリケーションがアプリケーション要件に詳述されている要件に準拠していることが重要である

2.2.2Import Session
[ファイル]メニューの[インポート...]オプションを使用して、nvprof の出力からインポートセッションを作成する
このオプションを選択すると、インポートプロセスをガイドするインポートダイアログが開く
実行可能アプリケーションはインポートセッションに関連付けられていないため、Visual Profiler はアプリケーションを実行して追加のプロファイルデータを収集できない
そのため、分析はインポートされたデータでのみ実行できる
また、GPU 詳細ビューにはインポートされたイベントとメトリック値が表示されるが、インポートセッションで新しいメトリックとイベントを選択して収集することはできない

2.2.2.1 シングルプロセス nvprof セッションのインポート
インポートダイアログを使用すると、複数の nvprof データファイルを新しいセッションにインポートできる
セッションのタイムライン情報を含む nvprof データファイルが 1 つ必要
このデータファイルは、-exportprofile オプションを指定した nvprof を実行して収集する必要がある
--system-profiling on などの他のオプションを有効にできるが、アプリケーションの実際の動作のタイムラインに影響するため、イベントやメトリックを収集しないこと
オプションで、アプリケーションのイベントおよびメトリック値を含む 1 つ以上のイベント/メトリックデータファイルを指定できる
これらのデータファイルは、-events および--metrics オプションのいずれかまたは両方を指定した nvprof を実行して収集する必要がある

分析システムに必要なすべてのイベントとメトリックを収集するには、-analysismetrics オプションを--kernels オプションと一緒に使用して、イベントとメトリックを収集するカーネルを選択する
詳細はリモートプロファイリングを参照のこと
セッションに複数の nvprof 出力ファイルをインポートする場合、アプリケーションがアプリケーション要件に準拠していることが重要

2.2.2.2 マルチプロセス nvprof セッションのインポート
インポートウィザードを使用すると、新しいマルチプロセスセッションにインポートする複数の nvprof データファイルを選択できる
各 nvprof データファイルには、いずれかのプロセスのタイムライン情報が含まれている必要がある
このデータファイルは、-export-profile オプションを指定して nvprof を実行することによって収集する必要がある
オプションで--system-profiling on などの他のオプションを有効にできるが、アプリケーションの実際の動作を表さないようにタイムラインを歪めるため、イベントやメトリックを収集しないこと
次の図に示すように、[Import the PGI Profiler Data]ダイアログで[Multiple Processes]オプションを選択する



複数のプロセスからタイムラインデータをインポートする場合、それらのプロセスのイベント/メトリックデータファイルを指定することはできない
マルチプロセスプロファイリングは、タイムラインデータに対してのみサポートされる

2.2.2.3 コマンドラインプロファイラセッションのインポート
コマンドラインプロファイラ(環境変数 COMPUTE_PROFILE を使用)のサポートは廃止されたが、以前のバージョンを使用して生成された CSV ファイルは引き続きインポートできる

インポートウィザードを使用するとコマンドラインプロファイラで生成された CSV ファイルを選択できる
複数の CSV ファイルをインポートすると、それらのコンテンツが結合され、単一のタイムラインに表示される
コマンドラインプロファイラ CSV ファイルは、gpustarttimestamp および streamid 構成パラメタを使用して生成する必要がある
イベントを含む他の構成パラメタを含めることは問題ない

2.3 アプリケーション要件
アプリケーションに関する性能データを収集するには、Visual Profiler がアプリケーションを確定的な方法で繰り返し実行できる必要がある
ソフトウェアとハードウェアの制限により、1 回のアプリケーション実行で必要なすべてのプロファイルデータを収集することはできない
アプリケーションが実行されるたびに、同じデータで動作し、同じカーネルとメモリのコピー呼び出しを同じ順序で実行する必要がある

具体的には
* デバイスの場合、コンテキスト作成の順序は、アプリケーションが実行されるたびに同じであること
各スレッドが独自のコンテキストを作成するマルチスレッドアプリケーションの場合、それらのコンテキスト作成の順序が複数の実行にわたって同じことを確認するように注意する必要があるたとえば、単一のスレッドでコンテキストを作成してから、そのコンテキストを他のスレッドに渡す必要がある場合がある
または、NVIDIA Tools Extension API を使用して、各コンテキストにカスタム名を提供する
アプリケーションの実行ごとに同じカスタム名が同じコンテキストに適用されれば、Visual Profiler は、複数の実行にわたってそれらのコンテキストを正しく関連付ける

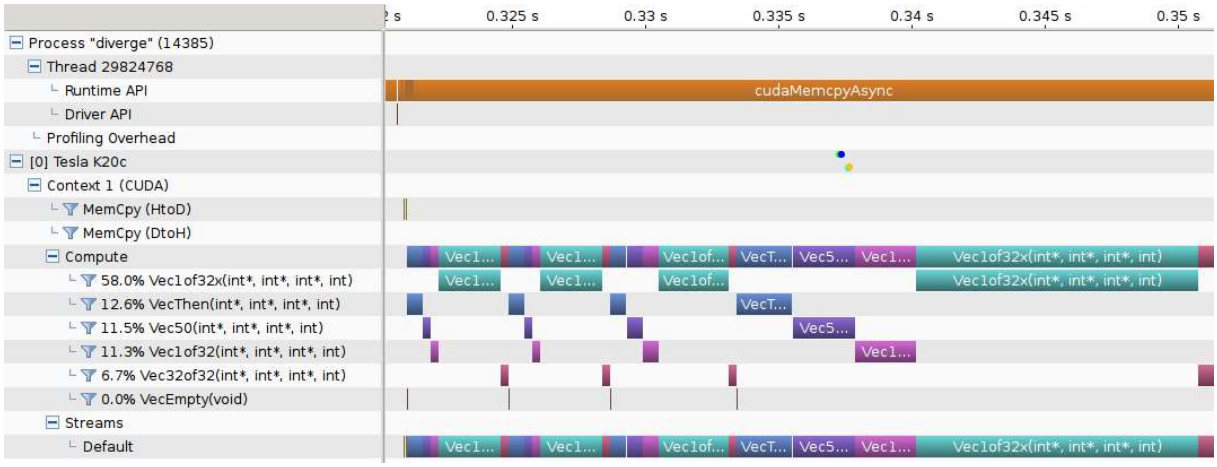
* コンテキストの場合、ストリーム作成の順序は、アプリケーションが実行されるたびに同一であること
または、NVIDIA Tools Extension API を使用して、各ストリームにカスタム名を提供する
アプリケーションを実行するたびに同じカスタム名が同じストリームに適用される限り、Visual Profiler はこれらのストリームを複数の実行にわたって正しく関連付ける

* ストリーム内では、カーネルと memcpy の呼び出し順序は、アプリケーションが実行されるたびに同一であること

2.4Visual Profiler ビュー
Visual Profiler はビューに編成される

ビューを一緒に使用すると、アプリケーションの性能を分析および視覚化できる
この章では、各ビューと、アプリケーションのプロファイリング中の使用方法を説明する

2.4.1Timeline View
Timeline View には、アプリケーションのプロファイリング中に発生した CPU と GPU のアクティビティが表示される
複数のタイムラインを同時に異なるタブで Visual Profiler で開く
次の図は、CUDA アプリケーションの Timeline View を示す



ビューの上部には、アプリケーションのプロファイリングの開始からの経過時間を示す水平ルーラーがある
ビューの左側には、タイムラインの水平方向の各行に何が表示されているかを説明する垂直ルーラーがあり、タイムラインのさまざまなコントロールが含まれる
これらのコントロールは、Timeline Controls 章で説明する

Timeline View はタイムライン行で構成される
各行は、行のタイプに対応するアクティビティの開始時間と終了時間を表す間隔を示す
たとえば、カーネルを表すタイムライン行には、そのカーネルの実行の開始時間と終了時間を表す
場合によっては(以下で説明するように)、タイムライン行にアクティビティの複数のサブ行が表示される
サブ行は、アクティビティが重複している場合に使用される
これらのサブ行は、重複するアクティビティの量に応じて、必要に応じて動的に作成される
特定のサブ行内に間隔を配置しても、特別な意味はない
間隔は、必要なサブ行の数を最小限にしようとするヒューリスティックを使用して、サブ行にパックされる
サブ行の高さは、垂直方向のスペースを適切に保つためにスケーリングされる

Timeline View に表示されるタイプは次のとおり

Process
タイムラインには、プロファイルされた各アプリケーションのプロセス行が含まれる
プロセス識別子は、プロセスの PID を表す
プロセスのタイムライン行には、アクティビティの間隔は含まれない
プロセス内のスレッドは、プロセスの子として表示される

Thread
タイムラインには、CUDA ドライバまたは CUDA ランタイム API 呼び出しのいずれかを実行したプロファイル対象アプリケーションの各 CPU スレッドのスレッド行が含まれる
スレッド識別子は、その CPU スレッドの一意の ID である
スレッドのタイムライン行には、アクティビティの間隔は含まれない

ランタイム API
タイムラインには、CUDA ランタイム API 呼び出しを実行する各 CPU スレッドのランタイム API 行が含まれる
行の各間隔は、対応するスレッドでの呼び出しの期間を表す

ドライバ API
タイムラインには、CUDA ドライバ API 呼び出しを実行する各 CPU スレッドのドライバ API 行が含まれる
行の各間隔は、対応するスレッドでの呼び出しの期間を表す

OpenACC
タイムラインには、OpenACC ディレクティブを呼び出す CPU スレッドごとに 1 つまたは複数の OpenACC 行が含まれる
行の各間隔は、対応するスレッドでの呼び出しの期間を表す
各 OpenACC タイムラインは複数の行で構成される場合がある
1 つのタイムライン内で、さらに下の行の OpenACC アクティビティは、上の行のアクティビティ内から呼び出される

OpenMP
タイムラインには、OpenMP を呼び出す CPU スレッドごとに 1 つの OpenMP 行が含まれる
行の各間隔は、アプリケーションが特定の OpenMP 領域または状態で費やす時間を表す
アプリケーションは同時に複数の状態になる可能性があるこれは、いくつかの間隔が重複する複数の行を描画することによって示される

pthread

ホストスレッド API 呼び出しが測定中に記録されている場合、タイムラインには、Pthread API 呼び出しを実行する CPU スレッドごとに 1 つの Pthread 行が含まれる
行の各間隔は、通話時間を表す
性能上の理由から、選択された Pthread API 呼び出しのみが記録されている場合があることに注意する

Markers and Ranges
タイムラインには、NVIDIA Tools Extension API を使用して時間範囲またはマーカーに注釈を付ける各 CPU スレッドの単一のマーカーおよび範囲行が含まれる
行の各間隔は、時間範囲の持続時間、またはマーカーの瞬間的なポイントを表す
範囲が重複している場合、この行にはサブ行がある

Profiling Overhead
タイムラインには、プロセスごとに 1 つのプロファイリングオーバーヘッド行が含まれる
行の各間隔は、プロファイリングに必要ないくつかのアクティビティの実行期間を表す
これらの間隔は、アプリケーションがプロファイルされないときに発生しないアクティビティを表す

Device
タイムラインには、プロファイリングされるアプリケーションによって利用される各 GPU デバイスのデバイス行が含まれる
タイムライン行の名前は、角かっこでデバイス ID を示し、その後にデバイスの名前が続きます
コンピューティング使用率分析を実行すると、行にはデバイスのコンピューティング使用率の推定値が含まれる
電力、クロック、および温度プロファイリングが有効になっている場合、行にはそれらの測定値を表すポイントも含まれる

Unified Memory
タイムラインには、統合メモリを使用する各 CPU スレッドおよびデバイスの統合メモリ行が含まれる
統合メモリには、CPU ページフォールト、GPU ページフォールト、データ移行(DtoH)、およびデータ移行(HtoD)の行が含まれる場合がある
セッションを作成するとき、ユーザーはユニファイドメモリタイムラインに対してセグメントモードまたは非セグメントモードを選択できる
セグメントモードでは、タイムラインが同じ幅のセグメントに分割され、各タイムセグメントの集計データ値のみが表示される
セグメントの数は変更できる
非セグメントモードでは、タイムラインの各間隔は収集された実際のデータを表し、各間隔のプ

ロパティを表示できる
セグメントは、ヒートマップカラースキームを使用して色付けされる
タイムラインのプロパティの下で、色の選択に使用されるプロパティが提供され、凡例には、プロパティ値のさまざまな範囲への色のマッピングが表示される

CPU Page Faults
これには、各 CPU スレッドの CPU ページフォールト行が含まれる
非セグメントモードでは、タイムラインの各間隔が 1 つの CPU ページ違反に対応する

Data Migration データ移行(DtoH)
タイムラインには、各デバイスのデータ移行(DtoH)行が含まれる
非セグメントモードでは、タイムラインの各間隔は、デバイスからホストへの 1 回のデータ移行に対応する

GPU ページ違反
タイムラインには GPU ページ違反が含まれる
各 CPU スレッドの行
非セグメントモードでは、タイムラインの各間隔が 1 つの GPU ページフォールトグループに対応する

Data Migration(HtoD)
タイムラインには、各デバイスのデータ移行(HtoD)行が含まれる
非セグメントモードでは、タイムラインの各間隔は、ホストからデバイスへの 1 つのデータ移行に対応する

Context
タイムラインには、GPU デバイス上の各 CUDA コンテキストのコンテキスト行が含まれる
タイムライン行の名前は、コンテキストに名前を付けるために NVIDIA Tools Extension API が使用された場合、コンテキスト ID またはカスタムコンテキスト名を示す
コンテキストの行には、アクティビティの間隔は含まれない

Memcpy
タイムラインには、memcpy を実行する各コンテキストのメモリコピー行が含まれる
コンテキストには、デバイスからホストへの、ホストからデバイスへの、デバイスからデバイスへの、およびピアツーピアのメモリコピーの最大 4 つの memcpy 行が含まれる場合がある
行の各間隔は、GPU で実行される memcpy の期間を表す

Compute

タイムラインには、GPU で計算を実行する各コンテキストの Compute 行が含まれる
行の各間隔は、GPU デバイス上のカーネルの期間を表す
Compute 行は、コンテキストのすべての計算アクティビティを示す
サブ行は、同時カーネルがコンテキストで実行されるときに使用される
CUDA Dynamic Parallelism を使用して起動されたカーネルを含むすべてのカーネルアクティビティは、コンピューティング行に表示される
Compute 行に続くカーネル行は、個々のアプリケーションカーネルのアクティビティを示す

Kernel
タイムラインには、アプリケーションによって実行される各カーネルの Kernel 行が含まれる
行の各間隔は、包含コンテキストでのそのカーネルのインスタンスの実行時間を表す
各行には、すべてのカーネルの合計実行時間と比較した、そのカーネルのすべてのインスタンスの合計実行時間を示すパーセンテージでラベルが付けられる
コンテキストごとに、カーネルはこの実行時間のパーセンテージによって上から下に並べられる
サブ行は、カーネルの同時実行を示すために使用される
CUDA Dynamic Parallelism アプリケーションの場合、カーネルは、カーネル間の親子関係を表す階層に編成される
ホスト起動カーネルは、コンテキスト行の直接の子として表示される
CUDA Dynamic Parallelism を使用して他のカーネルを起動するカーネルは、「+」アイコンを使用して展開し、それらの子カーネルを表す Kernel 行を表示できる
子カーネルを起動しないカーネルの場合、カーネル実行は実線の間隔で表され、カーネルのそのインスタンスが GPU で実行されていた時間を示す
子カーネルを起動するカーネルの場合、間隔には最後に中空部分を含めることもできる
中空部分は、カーネルが実行を終了した後、子カーネルの実行が終了するのを待っている時間を表す
CUDA 動的並列処理実行モデルでは、すべての子カーネルが完了するまで親カーネルが完了しないことが必要であり、これが中空部分が示しているものである
タイムラインコントロールで説明されているフォーカスコントロールを使用して、親/子タイムラインの表示を制御できる

Stream
タイムラインには、アプリケーションで使用される各ストリームのストリーム行が含まれる(デフォルトのストリームとアプリケーションで作成されたストリームの両方を含みます)
Stream 行の各間隔は、そのストリームで実行された memcpy またはカーネル実行の期間を表す

2.4.1.1 タイムラインコントロール
Timeline View には、タイムラインの表示方法を制御するために使用するいくつかのコントロールがある

これらのコントロールの一部は、GPU 詳細ビューと分析ビューでのデータの表示にも影響する

垂直タイムラインルーラーのサイズ変更
垂直ルーラーの幅は、ルーラーの右端にマウスポインタを置くことで調整できる
二重矢印ポインタが表示されたら、マウスの左ボタンをクリックして押したままドラッグする
垂直定規の幅はセッションとともに保存される

タイムラインの並べ替え
カーネルとストリームのタイムライン行を並べ替える
これらの行を並べ替えて、関連するカーネルとストリームを視覚化したり、重要でないカーネルとストリームをタイムラインの下部に移動したりできる
行を並べ替えるには、左クリックして行ラベルを押したままにする
二重矢印ポインタが表示されたら、上下にドラッグして行を配置する
タイムラインの順序はセッションとともに保存される

タイムラインのフィルタリング
Memcpy と Kernel の行をフィルタリングして、それらのアクティビティを GPU 詳細ビューと分析ビューの表示から除外できる
行を除外するには、行ラベルのすぐ左にあるフィルタアイコンを左クリックする
Kernel または Memcpy のすべての行をフィルタに掛けるには、Shift キーを押しながら左の行をクリックする
行がフィルタされると、その行の間隔は淡色表示され、フィルタされた状態を示す

タイムラインの拡大と縮小
タイムライン行のグループは、行ラベルのすぐ左にある[+]および[-]コントロールを使用して、展開および縮小できる
3 つの展開/折りたたみ状態がある
Collapsed-折りたたまれた行に含まれるタイムライン行は表示されない
Expanded-フィルタリングされないすべてのタイムライン行が表示される
All-Expanded-すべてのタイムライン行がフィルタされ、フィルタされない状態で表示される

折りたたまれた行に関連付けられた間隔は、それらのビューに設定されたフィルタモードによっては、GPU 詳細ビューと分析ビューに表示されない場合がある(詳細はビューのドキュメントを参照のこと)
たとえば、デバイスの行を折りたたむと、そのデバイスに関連付けられているすべての memcpy、memsets、およびカーネルが、これらのビューに表示される結果から除外される

タイムラインのカラーリング

タイムラインのカラーリングには 3 つのモードがある
カラーリングモードは、[View]メニュー、タイムラインコンテキストメニュー(Timeline View で右クリックしてアクセス)、およびプロファイラツールバーで選択できる
カーネルカラーリングモードでは、カーネルの各タイプに一意の色が割り当てられる(つまり、カーネル行のすべてのアクティビティインターバルは同じ色になる)
ストリームカラーリングモードでは、各ストリームに一意の色が割り当てられる(つまり、ストリームで発生するすべての memcpy およびカーネルアクティビティに同じ色が割り当てられる)
プロセスカラーリングモードでは、各プロセスに一意の色が割り当てられる(つまり、プロセスで発生するすべての memcpy とカーネルアクティビティに同じ色が割り当てられる)

Focusing Kernel Timelines カーネルのタイムラインに焦点を合わせる
CUDA 動的並列処理を使用するアプリケーションの場合、Timeline View には、カーネル間の親子関係を示すカーネルアクティビティの階層が表示される
デフォルトでは、すべての親子関係が同時に表示される
フォーカスタイムラインコントロールを使用して、表示されている親子関係を特定の限定された「家系図」のセットにフォーカスできる
フォーカスタイムラインモードは、タイムラインコンテキストメニュー(Timeline View で右クリックしてアクセス)、およびプロファイラツールバーで選択および選択解除できる

特定のカーネルの「ファミリーツリー」を表示すには、カーネルを選択してからフォーカスモードを有効にする
選択したカーネルの祖先または子孫以外のすべてのカーネルが非表示になる
フォーカスモードを有効にする前に、Ctrl-select を使用して複数のカーネルを選択できる
フォーカスモードを無効にしてすべてのカーネルを Timeline View に戻すには、「フォーカスしない」オプションを使用する

依存関係分析コントロール
タイムラインで依存関係の分析結果を視覚化するための 2 つのモードがある:
Focus Critical Path
Highlight Execution Dependencies
これらのモードは、[表示]メニュー、タイムラインコンテキストメニュー(Timeline View で右クリックしてアクセス)、および Visual Profiler ツールバーで選択できる
これらのオプションは、依存関係分析アプリケーション分析ステージが実行された後に使用可能になる(ガイドなしアプリケーション分析を参照)
これらのモードの詳細な説明は、Dependency Analysis Controls に記される

2.4.1.2 タイムラインのナビゲート
タイムラインは、アプリケーションの性能をよりよく理解して視覚化するために、いくつかの方

法でスクロール、ズーム、およびフォーカスできる

Zooming
ズームコントロールは、[表示]メニュー、タイムラインコンテキストメニュー(Timeline View で右クリックしてアクセス)、およびプロファイラツールバーで使用できる
ズームインするとビューに表示されるタイムスパンが短くなり、ズームアウトするとビューに表示されるタイムスパンが長くなるズームに合わせてビューが拡大縮小され、タイムライン全体が表示される
Ctrl キーを押しながらマウスホイールを使用してズームインおよびズームアウトできる(MacOSX の場合はコマンドキーを使用する)
もう 1 つの便利なズームモードは、領域ズームである
Ctrl キーを押しながら(MacOSX の場合はコマンドキーを使用)、マウスを左クリックしてドラッグし、タイムラインの領域を選択する
マウスボタンを離すと、ハイライト表示された領域が拡大され、ビュー全体が表示される

スクロール
タイムラインは、マウスホイールのスクロールバーで垂直にスクロールできる
タイムラインは、スクロールバーを使用するか、Shift キーを押しながらマウスホイールを使用して、水平方向にスクロールできる

ハイライト/相関
タイムラインのアクティビティ間隔の上にマウスポインタを移動すると、対応するアクティビティが表示されているすべての場所でその間隔が強調表示される
たとえば、カーネルの実行を表す間隔の上にマウスポインタを移動すると、そのカーネルの実行もストリームと計算タイムライン行で強調表示される
カーネルまたは memcpy 間隔が強調表示されると、対応するドライバまたはランタイム API 間隔も強調表示される
これにより、CPU でのドライバまたはランタイム API または OpenACC ディレクティブの呼び出しと GPU での対応するアクティビティとの相関関係を確認できる
強調表示された間隔に関する情報は、プロパティビューに表示される

選ぶ
タイムライン間隔または行を左クリックして選択できる
複数選択は、Ctrl-左クリックを使用する
間隔または行の選択を解除するには、もう一度 Ctrl キーを押しながら左クリックする
単一の間隔または行を選択すると、その間隔または行に関する情報がプロパティビューに固定される
GPU 詳細ビューでは、選択した間隔の詳細情報がテーブルに表示される

Measuring Time Deltas 時間デルタの測定
タイムライン上部の水平定規を左クリックしてドラッグすると、測定定規を作成できる
ルーラーが作成されたら、左クリックしてアクティブ化および非アクティブ化できる
Ctrl キーを押しながら左クリックすると、複数のルーラーをアクティブ化できる
定規はいくつでも作成できる
アクティブなルーラーは、Delete キーまたは Backspace キーで削除される
ルーラーを作成した後、タイムライン上に表示される垂直ガイドラインをドラッグすることで、ルーラーのサイズを変更できる
マウスがタイムライン間隔の上でドラッグされると、ガイドラインはその間隔の最も近い端にスナップする

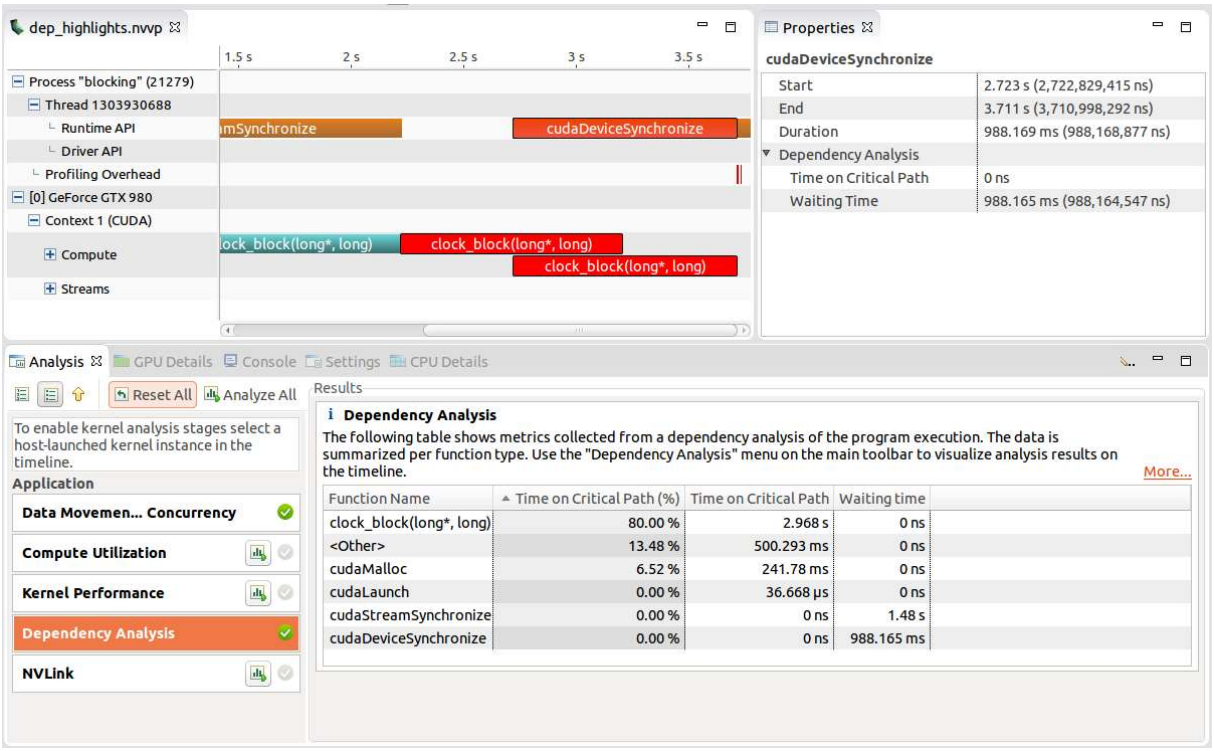
2.4.1.3 タイムラインの更新
プロファイラは、データを読み取るときに徐々にタイムラインを読み込みます
これは、ロードされるデータファイルが大きい場合、またはアプリケーションが大量のデータを生成した場合に、より明白になる
このような場合、タイムラインが部分的にレンダリングされることがある
同時に、回転する円が現在のセッションタブのアイコンに置き換わり、タイムラインが完全に読み込まれていないことを示す
アイコンが元に戻ったら、読み込みは完了する
メモリフットプリントを削減するために、プロファイラは、現在のズームレベルで表示されないタイムラインコンテンツの一部をロードしない場合がある
これらのコンテンツは、新しいズームレベルで表示されるときに自動的に読み込まれる

2.4.1.4 依存関係分析コントロール
プロファイラを使用すると、それぞれの分析ステージが実行されると、依存関係分析の結果をタイムラインで視覚化できる
依存関係分析の仕組みの詳細は「Dependency Analysis」を参照のこと

「フォーカスクリティカルパス」は、クリティカルパス上のすべての間隔に焦点を当て、他の間隔を弱めることにより、アプリケーション全体のクリティカルパスを視覚化する
モードが有効になっていて、タイムライン間隔が選択されている場合(左クリックすることにより)、選択された間隔がフォーカスされる
ただし、クリティカルパスは依然として中空の間隔として表示される
これにより、実行中のクリティカルパスを「追跡」し、個々の間隔を検査できる

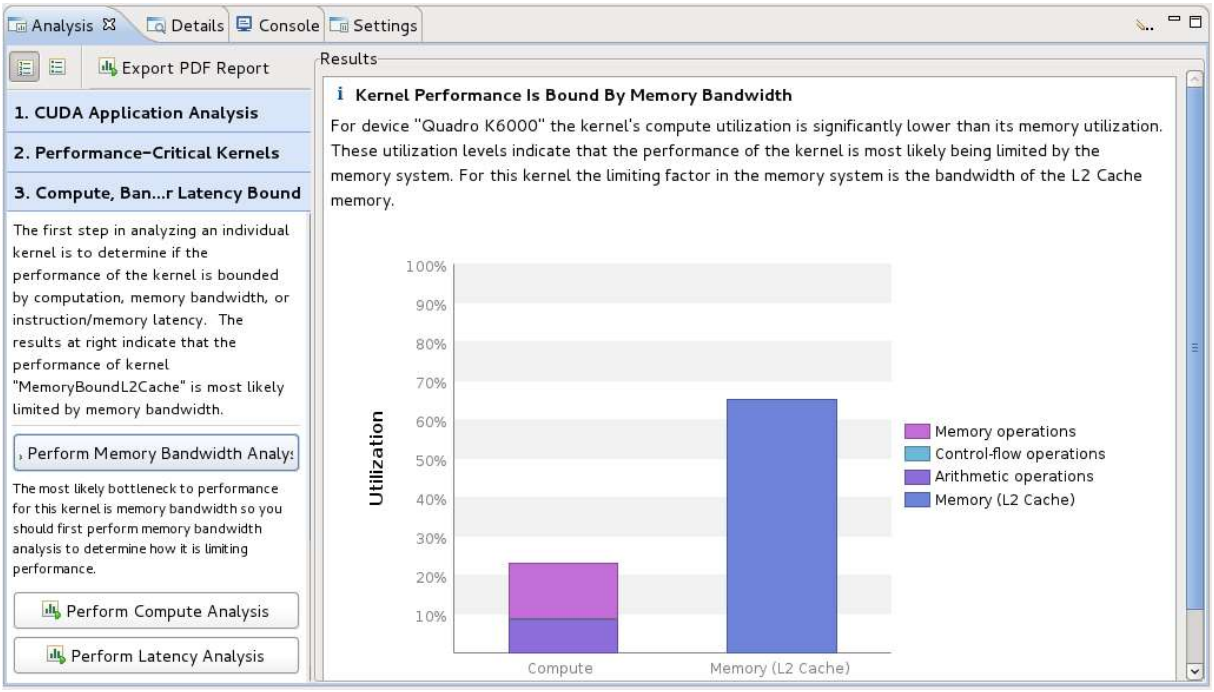
「実行依存関係の強調表示」を使用すると、各間隔の実行依存関係を分析できる(特定の間隔では、依存関係情報が収集されないことに注意する)

このモードを有効にすると、強調表示の色が黄色(相関間隔を表す)から赤(依存関係を表す)に変わる
選択された間隔とすべての着信および発信依存関係の両方が強調表示される



2.4.2 分析ビュー

分析ビューは、アプリケーション分析を制御し、分析結果を表示すために使用される
分析モードには、ガイド付きとガイドなしの 2 つがある
ガイドモードでは、分析システムが複数の分析段階をガイドし、アプリケーションで考えられる性能リミッタと最適化の機会を理解に役立つ
非ガイドモードでは、アプリケーション用に収集されたすべての分析結果を手動で探索できる
次の図は、ガイド付き分析モードの分析ビューを示す
ビューの左側には、アプリケーションの分析と最適化に役立つステップバイステップの指示が表示される
ビューの右側には、分析の各部分に適した詳細な分析結果が表示される



2.4.2.1 ガイド付きアプリケーション分析

ガイドモードでは、分析ビューは、アプリケーション内の各カーネルに提供される特定の分析ガイダンスと共に、アプリケーション全体の分析を段階的にガイドする

ガイド付き分析は「CUDA アプリケーション分析」から始まり、そこからアプリケーション内の最適化の機会に導きます

2.4.2.2 ガイドなしのアプリケーション分析

非ガイド分析モードでは、各アプリケーション分析ステージに[Run analysis]ボタンがあり、そのステージの分析結果を生成するために使用できる

「Run analysis」ボタンを選択すると、プロファイラがアプリケーションを実行して、分析の実行に必要なプロファイリングデータを収集する

分析ステージの横にある緑色のチェックマークは、そのステージの分析結果が利用可能であることを示す

各分析結果には、分析の簡単な説明と、分析に関する詳細なドキュメントへの[More...]リンクが含まれる

分析結果を選択すると、その結果に関連付けられているタイムライン行または間隔が Timeline View で強調表示される

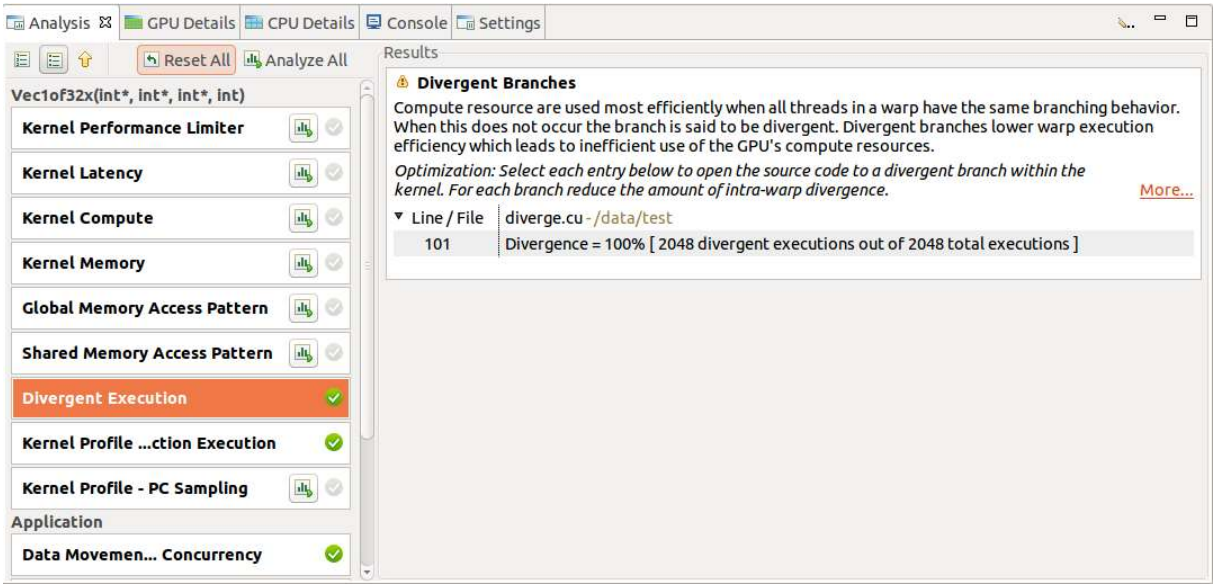
タイムラインで単一のカーネルインスタンスを選択すると、追加のカーネル固有の分析ステージを使用できる

カーネル固有の各分析ステージには、アプリケーション分析ステージの場合と同じように動作する[Run analysis]ボタンがある

次の図は、発散実行分析ステージの分析結果を示す

Divergent Execution などの一部のカーネルインスタンス分析結果は、カーネル内の特定のソース

行に関連付けられる
各結果に関連付けられているソースを表示すには、テーブルからエントリを選択する
そのエントリに関連付けられているソースファイルが開く

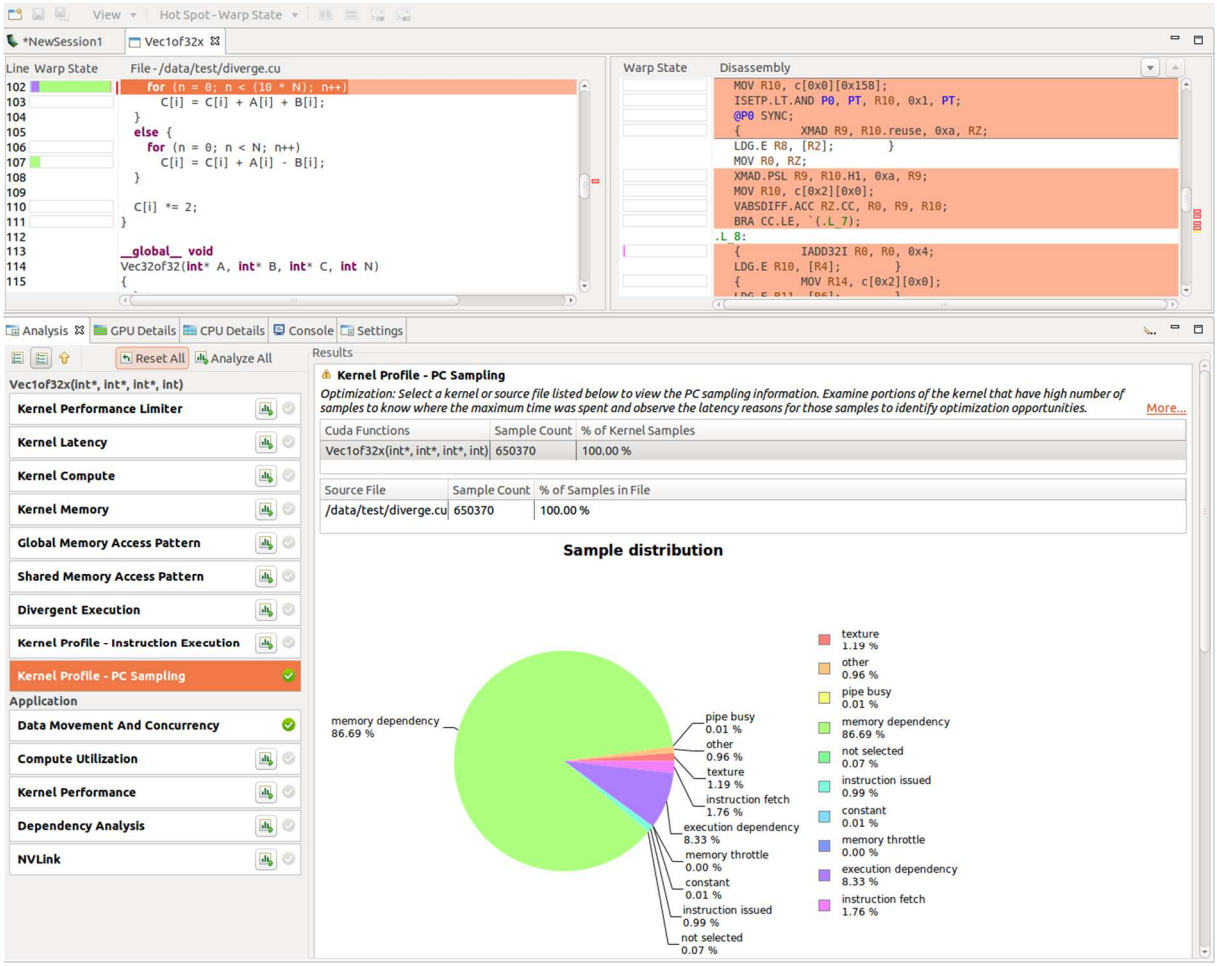


2.4.2.3 PC サンプリングビュー
コンピューティング機能が 5.2 以上のデバイス(モバイルデバイスを除く)には、PC サンプリングの機能がある
この機能では、PC とワープの状態は、SM ごとのアクティブなワープの 1 つについて定期的な間隔でサンプリングされる
ワープ状態は、そのワープがサイクルで命令を発行したかどうか、またはそれが停止して命令を発行できなかった理由を示す
サンプリングされたワープがストールすると、同じサイクルで他のワープが指示を出す可能性がある
したがって、サンプリングされたワープのストールは、必ずしも命令発行パイプラインに穴があることを示す必要はない
さまざまな状態の説明は、Warp State 章を参照のこと

コンピューティング機能 6.0 以降のデバイスには、レイテンシの理由を提供する新しい機能がある
レイテンシのサンプルは、問題のパイプラインに穴が開いている理由を示す
これらのサンプルを収集している間、それぞれのワープスケジューラでは命令が発行されないため、レイテンシの理由が示される
レイテンシの理由は、「選択されない」ストールの理由を除いて、ワープ状態章のストールの理由の 1 つになる

プロファイラはこの情報を収集し、「カーネルプロファイル-PC サンプリング」ビューに表示す

このビューでは、すべての関数とカーネルのサンプル分布が表に示される
円グラフは、各カーネルについて収集されたストール理由の分布を示す
ソースファイルまたはデバイス機能をクリックした後、カーネルプロファイル-PC サンプリングビューが開きます
垂直スクロールバーの横に表示されるホットスポットは、各ソースと組立ラインで収集されたサンプルの数によって決まります
ストール理由の分布は、各ソースおよび組立ラインの積み上げ棒として表示される
これは、ソースコードレベルでレイテンシの理由を特定に役立つ
コンピューティング機能 6.0 以上のデバイスの場合、Visual Profiler は 2 つのビューを表示する
ワープ状態ビューを提供する「Kernel Profile - PC Sampling」と、レイテンシの理由を提供する「Kernel Profile - PC Sampling - Latency」である
ホットスポットは、「ワープ状態」または「レイテンシ理由」のホットスポットを指すように選択できる
結果章の表は、合計レイテンシサンプル、発行パイプラインビジーサンプル、および命令発行サンプルのパーセント分布を示す
ブログ投稿の「命令レベルのプロファイリングによる性能の問題の特定」では、PC サンプリングを使用して CUDA カーネルを最適化する方法を示す
https://devblogs.nvidia.com/cuda-7-5-pinpoint-performance-problems-instruction-level-profiling/



2.4.2.4 メモリ統計

計算機能が 5.0 以上のデバイスには、カーネル実行中のメモリサブシステムの使用状況を表示する機能がある

このグラフは、CUDA プログラミングモデルのメモリ階層の概要を示す

図の緑色のノードは論理メモリ空間を示し、青色のノードはチップ上の実際のハードウェアユニットを示す

さまざまなキャッシュについて、報告されるパーセンテージの数値はキャッシュヒット率を示すこれは、行われたすべてのリクエストに対する、キャッシュでローカルに利用可能なデータで処理できるリクエストの比率である

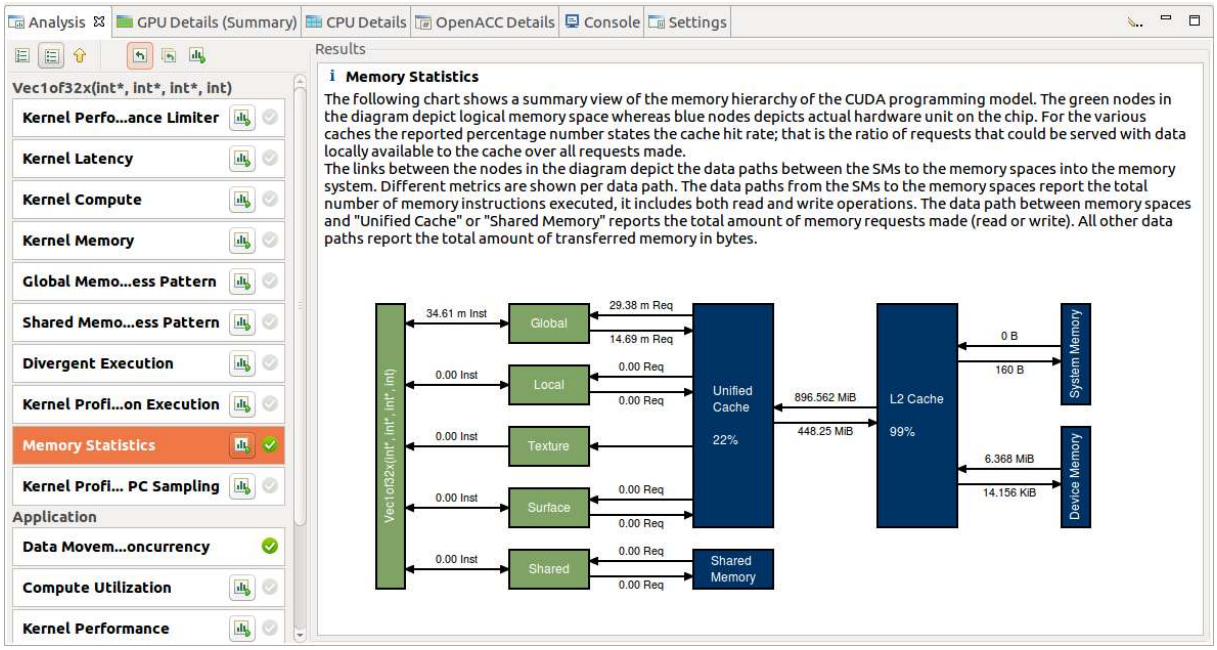
図中のノード間のリンクは、SM からメモリシステムへのメモリ空間へのデータパスを示すデータパスごとに異なるメトリックが表示される

SM からメモリスペース(グローバル、ローカル、テクスチャ、サーフェス、共有)へのデータパスは、実行されたメモリ命令の総数を報告するこれには、読み取り操作と書き込み操作の両方が含まれる

メモリ空間と「Unified Cache」または「Shared Memory」間のデータパスは、行われたメモリ要求の合計量を報告する

他のすべてのデータパスは、転送されたメモリの総量をバイト単位で報告する

右方向を指す矢印は書き込み操作を示し、左方向を指す矢印は読み取り操作を示す



2.4.2.5 NVLink ビュー

NVIDIA NVLink は、CPU と GPU 間、および GPU 間の高速通信を可能にする、高帯域幅でエネルギー効率の高い相互接続である

Visual Profiler は、NVLink トポロジと NVLink 送信/受信スループットメトリックを収集し、メトリックをトポロジにマップする

トポロジは、デフォルトでタイムラインとともに収集される

スループット/使用率メトリックは、NVLink オプションが選択されている場合にのみ生成される

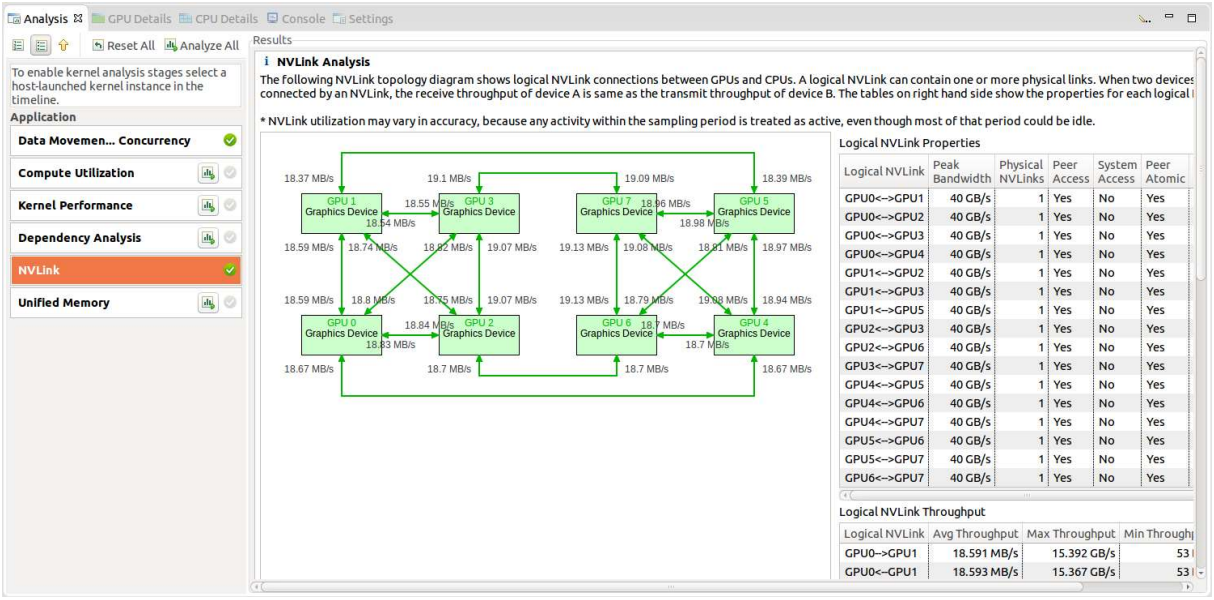
NVLink 情報は、ガイド付き分析の CUDA アプリケーション分析における GPU 使用状況の調査の結果章に表示される

NVLink 分析は、異なるデバイス間の論理的な NVLink 接続を示すトポロジを示す

論理リンクは、2 つのデバイス間に接続された同じプロパティの 1-4 の物理 NVLink で構成される

Visual Profiler は、論理 NVLink のプロパティと達成された使用率を「論理 NVLink プロパティ」テーブルにリストする

また、「論理 NVLink スループット」テーブルに論理 NVLink の送信および受信スループットを示す



2.4.3 ソース分解ビュー

Source-Disassembly ビューは、ソースおよびアセンブリ命令レベルでカーネルの分析結果の表示に使用される

カーネルソースを表示すには、「-lineinfo」オプションを使用してコードをコンパイルする必要がある

このコンパイラオプションを使用しない場合は、逆アセンブリビューのみが表示される

このビューは、次のタイプの分析で表示される

*グローバルメモリアクセスパターン分析

*共有メモリアクセスパターン分析

*発散実行分析

*カーネルプロファイル-命令実行分析

*カーネルプロファイル-PC サンプリング分析

カーネルの「ガイド付き分析」または「ガイドなし分析」の一部として、分析結果が分析ビューに表示される

ソースファイルまたはデバイス機能をクリックすると、Source-Disassembly ビューが開きます

ソースファイルが見つからない場合は、ダイアログが開いて、ソースファイルの新しい場所を選択してポイントする

これは、たとえば、別のシステムでプロファイリングが行われたときに発生する可能性がある

Source-Disassembly ビューには以下が含まれる

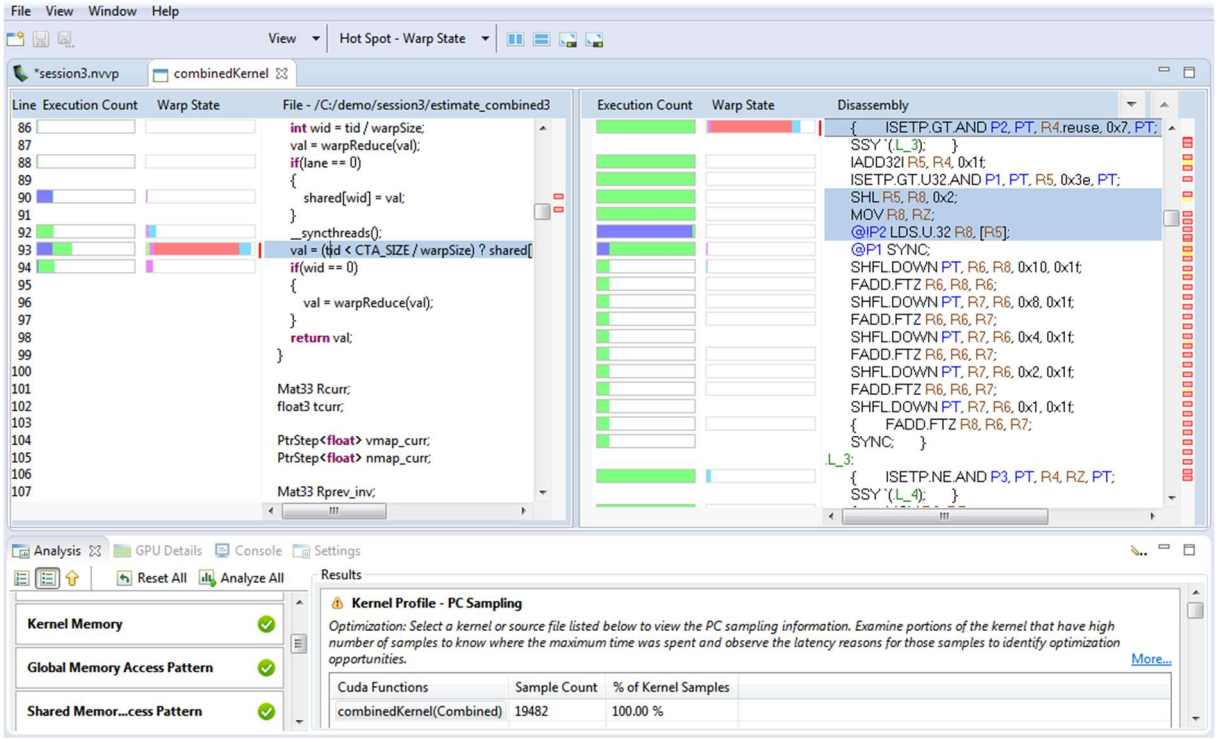
*高レベルのソース

*組立説明

*ソースレベルのホットスポット

*組立指示レベルのホットスポット

*ソースレベルに集約されたプロファイリングデータの列
*アセンブリ命令レベルで収集されたプロファイリングデータの列



Source-Disassembly ビューに表示される情報は、次のツールバーオプションでカスタマイズできる
*[View]メニュー - 利用可能なプロファイラデータ列から 1 つ以上を選択する
これは、分析タイプに基づいてデフォルトで選択される
*ホットスポットメニュー - ホットスポットに使用するプロファイラデータを選択する
これは、分析タイプに基づいてデフォルトで選択される
*ソースビューと逆アセンブリビューを並べて表示す
*ソースビューと逆アセンブリビューを上から下に表示す
*ソースビューを最大化する
*分解ビューを最大化

ホットスポットは、重要度のレベル(低、中、高)に基づいて色分けされる
ホットスポットの上にマウスを置くと、プロファイラデータの値、重要度のレベル、ソースまたは逆アセンブリラインが表示される
ソースレベルまたはアセンブリ命令レベルでホットスポットをクリックして、ホットスポットに対応するソースまたは分解行を表示できる

逆アセンブリビューでは、選択したソース行に対応するアセンブリ命令が強調表示される
逆アセンブリ列ヘッダの右側に表示される上矢印ボタンと下矢印ボタンをクリックして、次または前の命令ブロックに移動できる

2.4.4 GPU 詳細ビュー

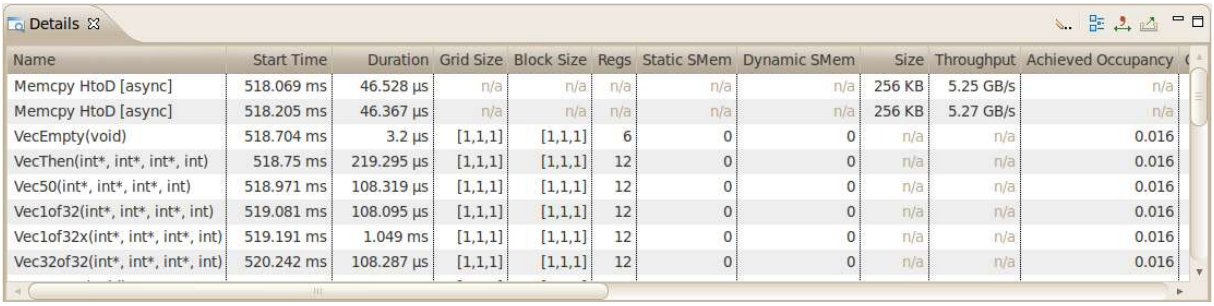
GPU 詳細ビューには、プロファイルされたアプリケーションでの各メモリコピーとカーネル実行に関する情報の表が表示される

次の図は、いくつかの memcpy とカーネルの実行を含むテーブルを示す

テーブルの各行には、カーネルの実行またはメモリのコピーに関する一般的な情報が含まれる

カーネルの場合、テーブルには、そのカーネルについて収集された各メトリックまたはイベント値の列も含まれる

図の「Achieved Occupancy 達成された占有率」列は、カーネル実行ごとのそのメトリックの値を示す


Details

Name	Start Time	Duration	Grid Size	Block Size	Regs	Static SMem	Dynamic SMem	Size	Throughput	Achieved Occupancy
Memcpy HtoD [async]	518.069 ms	46.528 µs	n/a	n/a	n/a	n/a	n/a	256 KB	5.25 GB/s	n/a
Memcpy HtoD [async]	518.205 ms	46.367 µs	n/a	n/a	n/a	n/a	n/a	256 KB	5.27 GB/s	n/a
VecEmpty(void)	518.704 ms	3.2 µs	[1,1,1]	[1,1,1]	6	0	0	n/a	n/a	0.016
VecThen(int*, int*, int*, int)	518.75 ms	219.295 µs	[1,1,1]	[1,1,1]	12	0	0	n/a	n/a	0.016
Vec50(int*, int*, int*, int)	518.971 ms	108.319 µs	[1,1,1]	[1,1,1]	12	0	0	n/a	n/a	0.016
Vec1of32(int*, int*, int*, int)	519.081 ms	108.095 µs	[1,1,1]	[1,1,1]	12	0	0	n/a	n/a	0.016
Vec1of32x(int*, int*, int*, int)	519.191 ms	1.049 ms	[1,1,1]	[1,1,1]	12	0	0	n/a	n/a	0.016
Vec32of32(int*, int*, int*, int)	520.242 ms	108.287 µs	[1,1,1]	[1,1,1]	12	0	0	n/a	n/a	0.016

列ヘッダを左クリックして列ごとにデータを並べ替えたり、列ヘッダを左クリックして新しい場所にドラッグしたりして、列を並べ替える

テーブルの行を選択すると、対応する間隔が「Timeline View」で選択される

同様に、Timeline View でカーネルまたは memcpy の間隔を選択すると、テーブルがスクロールされ、対応するデータが表示される

列ヘッダの上にマウスを置くと、ツールチップにその列に表示されているデータが表示される

イベントまたはメトリックデータを含む列の場合、ツールチップは対応するイベントまたはメトリックを説明する

[Metrics Reference]章には、各メトリックに関する詳細情報が含まれる

GPU 詳細ビューに表示される情報は、[詳細ビュー]ツールバーからアクセスできるメニューを使用して、さまざまな方法でフィルタリングできる

次のモードを使用できる

*Filter By Selection-選択すると、GPU 詳細ビューには、選択したカーネルと memcpy 間隔のデータのみが表示される

*非表示のタイムラインデータを表示-選択しない場合、データは、タイムラインに表示されているカーネルと memcpy についてのみ表示される

タイムラインの折りたたまれた部分の内側にあるため表示されないカーネルと memcpy は表示されない

*フィルタされたタイムラインデータを表示-選択されない場合、フィルタされないタイムライン行にあるカーネルと memcpy のデータのみが表示される

イベントとメトリックの収集
カーネルごとに特定のイベントとメトリックの値を収集して、詳細テーブルに表示できる
ビューの右上隅にあるツールバーアイコンを使用して、各デバイスについて収集するイベントとメトリックを構成し、アプリケーションを実行してそれらのイベントとメトリックを収集する

要約データを表示
デフォルトでは、テーブルには memcpy とカーネルの呼び出しごとに 1 行が表示される
または、表に各カーネル関数の要約結果を表示する
ビューの右上隅にあるツールバーアイコンを使用して、サマリー形式を選択または選択解除する

テーブルコンテンツのフォーマット
テーブル内の数値は、グループ化区切り文字の有無にかかわらず表示できる
グループ化区切りを選択または選択解除するには、ビューの右上隅にあるツールバーアイコンを使用する

詳細のエクスポート
ビューの右上隅にあるツールバーアイコンを使用して、テーブルの内容を CSV 形式でエクスポートできる

2.4.5 CPU 詳細ビュー

CPU 詳細ビュー
このビューは、アプリケーションが CPU で関数の実行に費やす時間の詳細を示す
各スレッドは、そのコールスタックをキャプチャするために定期的にサンプリングされ、これらの測定の概要がこのビューに表示される
トップダウン、ボトムアップ、コード構造(3)、表示スレッドの選択(1)、特定のスレッドのソートまたはハイライト(7、8)により、コールスタックを整理するためのさまざまな指示でビューを操作する

1.「all threads」オプションが選択されている場合(デフォルト)、プロファイルされたすべてのスレッドが 1 つのビューに表示される
このドロップダウンメニューを使用して個々のスレッドを選択できる

2.この列には、CPU でのアプリケーション実行の構造を表すイベントのツリーが表示される
残りの各列は、このイベントについて収集された測定値を示す
ここに表示されるイベントは、選択されているツリー方向モードによって決定される(3)

3.ツリーは、関数間の呼び出し階層を示すように編成される
次のモードを使用できる
*トップダウン(呼び出し元が最初)の呼び出しツリービュー-CPU 詳細ツリーは呼び出しツリーとして編成され、各関数は呼び出し元の子として表示される
このモードでは、 'main'関数から始まる呼び出しスタックを確認できる
*ボトムアップ(呼び出し先が最初)の呼び出しツリービュー-CPU の詳細ツリーは、各関数が呼び出す関数の子として表示されるように編成される
このモードでは、アプリケーションの実行に最も時間を費やしている呼び出しパスをすばやく特定できる
*コード構造(ファイルと行)のツリービュー-CPU の詳細ツリーには、各ソースファイルとライブラリに属する関数と、アプリケーションの実行が特定のソースコード行に起因する量が表示される
すべてのモードで、各関数にリストされた時間は「包括的」であり、この関数とそれが呼び出す関数の両方で費やされた時間を含みます
コード構造ビューの場合、コードの領域は包括的である(つまり、ファイルエントリは、ファイル内に含まれるすべての関数で費やされた時間をリストする)

4.この列には、このイベントであるべてのスレッドが費やした合計時間が、すべてのイベントで費やされた合計時間の割合として表示される

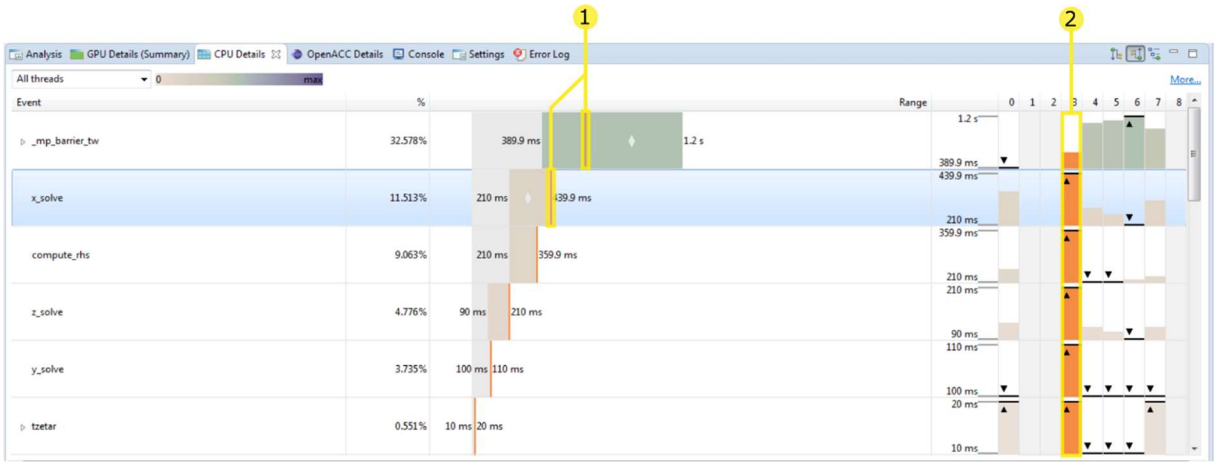
5.この列には、任意のスレッドがイベントに費やした時間が常にこの範囲内である範囲を示すバーが表示される
左側には最小値、右側には最大値が記載される
また、スペースがある場合は、バーの中央に小さな「ダイヤモンド」が描かれ、すべてのスレッドでこのイベントに費やされる平均時間が示される

6.これらの列には、イベントごとに異なるグラフが表示される
左側は縦軸の目盛りで、範囲グラフに示されているものと同じ最小値と最大値を示す
以下の列はそれぞれ、このイベントでスレッドによって費やされた時間を示す
特定のイベントとスレッドの組み合わせのセルがグレー表示されている場合、このイベントでこのスレッドが費やした時間はない(この例では、スレッド 1 と 2 の両方がイベント'x_solve'に時間を割かない)
さらに、すべてのスレッド間でイベントに費やされた最小または最大の時間を持つスレッドには、「三角形/線」の注釈が付けられる
この例では、イベント「x_solve」ではスレッド 3 が最も多く、スレッド 6 が最も短い時間を割いた

7.特定のスレッドで費やした時間で行を並べ替えるには、スレッドの列ヘッダをクリックする

8.特定のスレッドを強調表示すには、このチャートの棒の 1 つをクリックする

このビューへの変更は、スレッド 3(7)による並べ替えと強調表示(8)の結果である



1.スレッド 3 を強調表示すると、範囲チャートに垂直線が表示され、このスレッドがこのイベントに費やした時間を、すべてのスレッドの範囲と比較して示す
2.このスレッドも各行で強調表示される

CPU スレッド

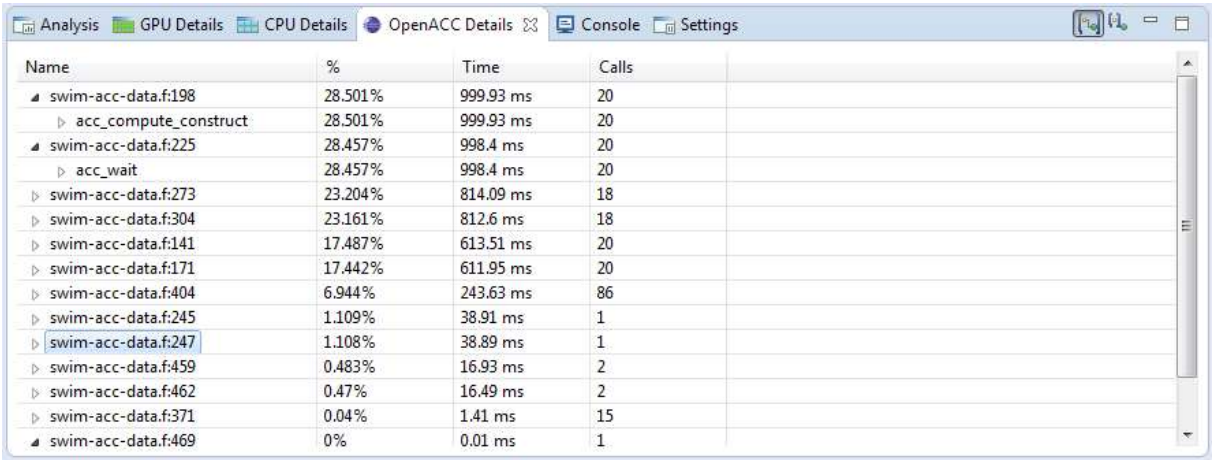
CPU ソースコード
ツリーでダブルクリックすると、任意の関数の CPU ソースビューが開く
表示するには、ソースファイルがローカルファイルシステム上にある必要がある
デフォルトでは、実行可能ファイルまたはプロファイルファイルを含むディレクトリが検索される
ソースファイルが見つからない場合は、その場所を尋ねるプロンプトが表示される
特定のディレクトリ内のファイルが検索される場合があるこの場合、このディレクトリが存在する場所へのパスを指定する必要がある

ヒント:
CPU プロファイルは、実行中のアプリケーションの状態を定期的にサンプリングすることによって収集される
このため、実行中にサンプリングされた関数のみがこのビューに表示される
実行時間の短い関数や頻繁に呼び出されない関数は、サンプリングされる可能性が低くなる
関数がサンプリングされなかった場合、その関数が実行されていた時間は、その関数を呼び出した関数に起因する
アプリケーションの性能を表す CPU プロファイルを収集するには、対象のコードを実行して、十分なサンプルを収集する必要がある
通常、1 分の実行時間で十分である

ヒント:
ファイルと行の情報は、コンパイラが取得したアプリケーションのデバッグ情報から収集される
この情報を確実に利用できるようにするには、 '-g'または同様のオプションを使用してコンパイルすることを推奨する

2.4.6 OpenACC 詳細ビュー

OpenACC テーブルビュー



OpenACC 詳細ビューには、プロファイルされたアプリケーションによって実行された各 OpenACC ランタイムアクティビティが表示される
各アクティビティはソースの場所でグループ化される：
つまりアプリケーションのソースコードで同じファイルと行番号で発生する
各アクティビティは、ソースの場所でラベル付けされたノードの下に配置される
各アクティビティは、プロファイルされたアプリケーションが費やした時間を、時間の単位と、このアプリケーションが OpenACC アクティビティを実行していた合計時間の割合の両方として示す
また、このアクティビティが呼び出された回数も表示される

特定の OpenACC アクティビティに費やされた時間をカウントするには、2 つの方法がある
* Show the Inclusive durations (counting any other OpenACC activities running at the same time) in the OpenACC details view
OpenACC の詳細ビューに包括期間を表示する(同時に実行されている他の OpenACC アクティビティをカウントしたもの)
OpenACC の詳細ビューには、このアクティビティの結果として実行されたアクティビティを含む、各アクティビティで費やされた合計時間が表示される
この場合、特定のアプリケーションソースの場所で発生する各アクティビティに費やされた時間の合計が、ソースの行に表示される

* Show the Exclusive durations (excluding any other OpenACC activities running at the same time) in the OpenACC details view
OpenACC 詳細ビューに排他期間を表示する(同時に実行されている他の OpenACC アクティビティを除く)
OpenACC の詳細ビューに、特定のアクティビティでのみ費やされた時間が表示される
この場合、特定のソースの場所で費やされる時間は常にゼロ時間であり、このソースの場所で発生する各アクティビティにのみ起因する

2.4.7 OpenMP 詳細ビュー

OpenMP テーブルビュー

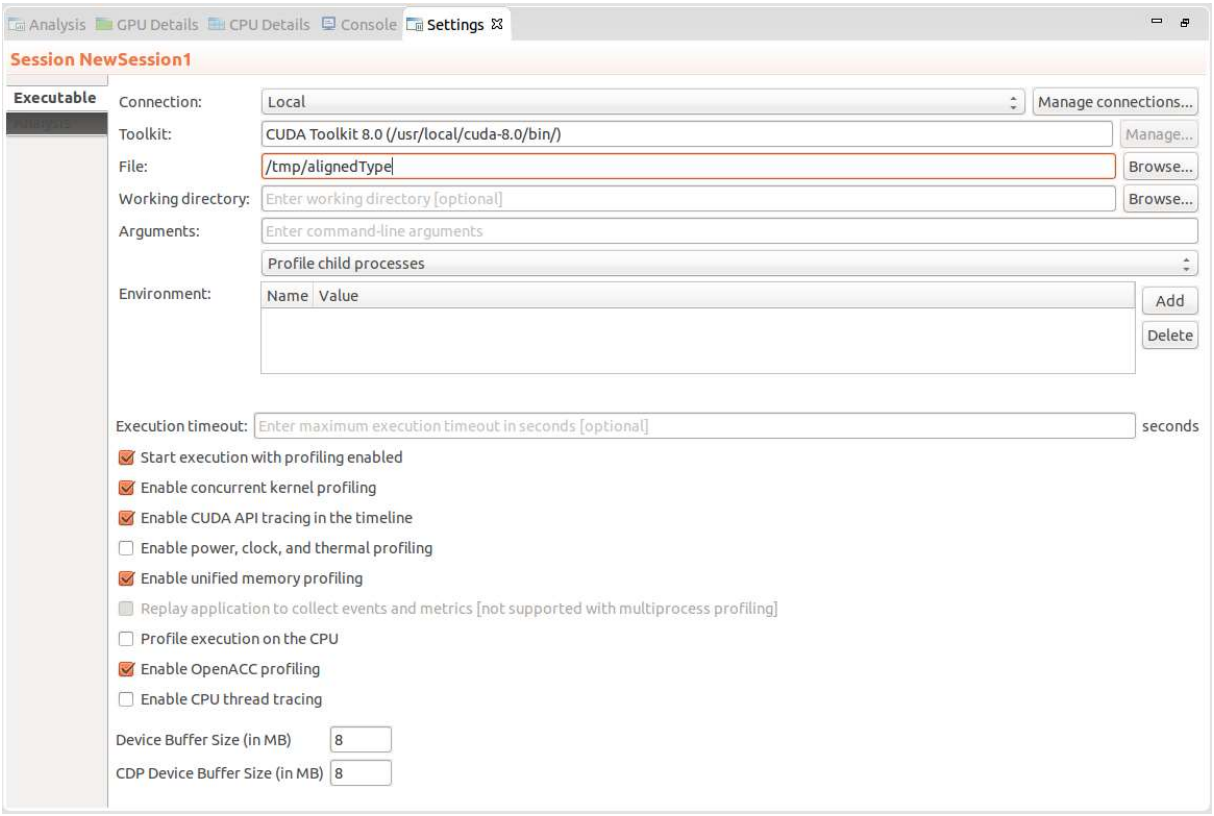
OpenMP の詳細ビューには、CPU での OpenMP ランタイムのアクティビティが表示される
アプリケーションが並列領域またはアイドリングで費やす時間は、タイムラインとこのビューの

両方に表示される
各タイプのアクティビティに費やされた時間の割合の基準は、最初の並列領域の開始から最後の並列領域の終了までの時間である
OpenMP ランタイムは同時に複数の状態になる可能性があるため、各アクティビティタイプのパーセンテージの合計は 100%を超えることがよくある

2.4.8 プロパティビュー
プロパティビューには、Timeline View で強調表示または選択された行または間隔に関する情報が表示される
行または間隔が選択されない場合、表示される情報はマウスポインタの動きを追跡する
行または間隔が選択されている場合、表示される情報はその行または間隔に固定される
関連付けられたソースファイルを持つ OpenACC 間隔が選択されると、このファイル名はソースファイルテーブルエントリに表示される
ファイルシステムで使用可能な場合、ファイル名をダブルクリックすると、対応するソースファイルが開きます

2.4.9 コンソールビュー
「コンソールビュー」には、アプリケーションを実行するたびに、アプリケーションの stdout および stderr 出力が表示される
アプリケーションに stdin 入力を提供する必要がある場合は、コンソールビューに入力する

2.4.10Settings View 設定ビュー
「Settings View」では、プロファイルされるアプリケーションの実行設定を指定する
次の図に示すように、「 Executable」タブでは、実行可能ファイル、作業ディレクトリ、コマンドライン引数、およびアプリケーションの環境を指定する
実行可能ファイルのみが必須で、他のすべてのフィールドはオプションである



Exection Timeout 実行タイムアウト

「Executable settings」タブでは、オプションの実行タイムアウトを指定できる

実行タイムアウトが指定されている場合、その秒数後にアプリケーションの実行が終了する

実行タイムアウトが指定されない場合、アプリケーションは正常に終了するまで実行を継続する

タイマーは、CUDA ドライバが初期化された瞬間からカウントを開始する

アプリケーションが CUDA API を呼び出さない場合、タイムアウトはトリガされない

プロファイリングを有効にして実行を開始する

「Enable concurrent kernel profiling」チェックボックスはデフォルトで設定され、アプリケーションのプロファイリングがアプリケーション実行の開始時に開始することを示す

cudaProfilerStart()および cudaProfilerStop()を使用して、「フォーカスされたプロファイリング」で説明されているようにアプリケーション内のプロファイリングを制御している場合は、このボックスをオフにする必要がある

同時カーネルプロファイリングを有効にする

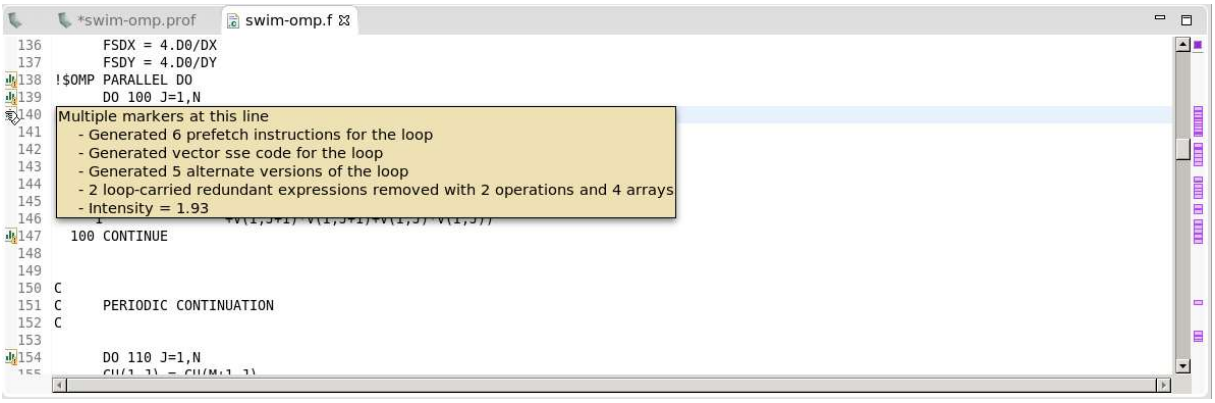
「Enable concurrent kernel profiling」チェックボックスはデフォルトで設定されており、カーネルの同時実行を利用するアプリケーションのプロファイリングを有効にする

このチェックボックスが設定されない場合、プロファイラはカーネルの同時実行を無効にする

カーネルの同時実行を無効にすると、プロファイリングのオーバーヘッドを削減できる場合があるため、同時カーネルを利用しないアプリケーションに適している場合がある

電力、クロック、熱プロファイリングを有効にする
「Enable power, clock, and thermal profiling」チェックボックスを設定して、アプリケーションで使用される各 GPU の電力、クロック、および熱動作の低周波数サンプリングを有効にする

2.4.11 CPU ソースビュー



CPU ソースコードビューでは、プロファイルされたアプリケーションの CPU ソースを構成するファイルを検査できる
このビューは、ツリー内の関数をダブルクリックすることにより、CPU の詳細ビューで開くこの関数に対応するソースファイルが開かれます
行番号は、左側のルーラーを右クリックして有効にできる

PGIR コンパイラを使用してコンパイルする場合、このビューに注釈を追加できる(詳細は Common Compiler Feedback Format を参照のこと)
これらの注釈は、特定のコード行がどのようにコンパイルされるかについてのメモである
PGI コンパイラは、プログラムがどのように最適化されたか、または特定の最適化が行われなかった理由に関する情報を保存する
これを CPU の詳細ビューと組み合わせて、特定のコード行がなぜそのように実行されたかを特定に役立つ
たとえば、次のようなメッセージが表示される場合がある
* コンパイラによって生成されるベクトル命令
* ループの計算強度、メモリ操作に対する比率の計算値が大きいほど、メモリのロードとストアよりも多くの計算が行われることを意味する
* 並列化に関する情報コンパイラが自動並列化できなかった場合に、ループを並列実行できるようにするためのヒントが含まれる

2.5 プロファイラのカスタマイズ

Visual Profiler を初めて起動したとき、およびようこそページを閉じた後、ビューのデフォルトの配置が表示される
ビューを移動してサイズを変更することで、開発ニーズに合わせてプロファイラをカスタマイズ

できる
加えた変更は、次回プロファイラを起動したときに復元される

2.5.1 ビューのサイズ変更
ビューのサイズを変更するには、ビュー間の分割領域を左クリックしてドラッグする
1 つの領域に積み重ねられたすべてのビューは、同時にサイズ変更される

2.5.2 ビューの並べ替え
スタックされたビューのセットでビューを並べ替えるには、ビュータブを左クリックして、ビュースタック内の新しい場所にドラッグする

2.5.3 ビューの移動
ビューを移動するには、ビュータブを左クリックして、新しい場所にドラッグする
ビューをドラッグすると、アウトラインにビューのターゲット位置が表示される
ビューを新しい場所に配置するか、他のビューと同じ場所に重ねる

2.5.4 ビューのドッキング解除
ビューをプロファイラウィンドウから切り離して、ビューが独自のスタンドアロンウィンドウを占有するようにする
これを行うと、複数のモニターを利用したり、個々のビューのサイズを最大化したりできる
ビューのドッキングを解除するには、ビュータブを左クリックして、プロファイラウィンドウの外にドラッグする
ビューをドッキングするには、ビューのタブ(ウィンドウの装飾ではない)を左クリックして、プロファイラウィンドウにドラッグする

2.5.5 ビューを開く／閉じる
ビューを閉じるには、ビュータブの X アイコンを使用する
ビューを開くには、[表示]メニューを使用する

2.6 コマンドライン引数
コマンドラインから Visual Profiler を起動すると、コマンドライン引数を使用して、次のパターンのいずれかを使用して、nvprof からエクスポートされた新しいセッションを開始するか、プロファイルファイルをインポートする実行可能ファイルを指定できる
* 実行可能ファイルの名前で nvvp を起動し、オプションで引数を指定して、新しい実行可能セッションを開始する

    nvvp executableName [[executableArguments] ...]

* 単一の「.nvprof」ファイルを引数として nvvp を起動して、単一プロセス nvprof セッションをインポートする(詳細は「nvprof's export/import options 」章を参照のこと)

nvvp data.nvprof

* 複数の「.nvprof」ファイルを引数として nvvp を起動して、マルチプロセス nvprof セッションをインポートする

nvvp data1.nvprof data2.nvprof ...

訳 佐賀大学 今井康貴 imaiy@cc.saga-u.ac.jp