第 3 章 NVPROF

nvprof プロファイリングツールを使用すると、コマンドラインからプロファイリングデータを収集して表示できる
nvprof を使用すると、カーネル実行、メモリ転送、メモリセット、CUDA API 呼び出し、CUDA カーネルのイベントやメトリックなど、CPU と GPU の両方での CUDA 関連のアクティビティのタイムラインを収集する
プロファイリングオプションは、コマンドラインオプションを介して nvprof に提供される
プロファイリング結果は、プロファイリングデータが収集された後にコンソールに表示され、nvprof または Visual Profiler で後で表示すために保存する

プロファイラのテキスト出力は、デフォルトで stderr にリダイレクトされる
--logfile を使用して、出力を別のファイルにリダイレクトする
出力のリダイレクトを参照のこと

コマンドラインからアプリケーションをプロファイルするには:
　nvprof [オプション] [アプリケーション] [アプリケーション引数]

完全なヘルプページを表示すには、nvprof --help と入力する

3.1 コマンドラインオプション

3.1.1. CUDA プロファイリングオプション

aggregate-mode
集約モード
後続の-events および--metrics オプションで指定されたイベントおよびメトリックの集約モードをオン/オフにする
これらのイベント/メトリック値は、デバイス全体ではなく、ドメインインスタンスごとに収集される
詳細は Event/metric Trace Mode を参照のこと

analysis-metrics
分析指標
Visual Profiler の「analysis」モードにインポートするプロファイリングデータを収集する
注:出力ファイルを指定するには、-export-profile を使用する

annotate-mpi
MPI 呼び出しに NVTX マーカーで自動的に注釈を付ける
マシンにインストールされている MPI 実装を指定する

現在、Open MPI および MPICH の実装がサポートされる
詳細は Automatic MPI Annotation with NVTX を参照のこと

concurrent-kernels
同時カーネル
カーネルの同時実行をオン/オフにする
カーネルの同時実行がオフの場合、1 つのデバイスで実行されているすべてのカーネルがシリアル化される

continuous-sampling-interval
連続サンプリング間隔
連続モードのサンプリング間隔をミリ秒単位で設定する
最小値は 1 ミリ秒である

cpu-thread-tracing
CPU スレッド API アクティビティに関する情報を収集する
詳細は CPU スレッドトレースを参照のこと

dependency-analysis
依存関係の分析
ホストとデバイスのアクティビティのイベント依存関係グラフを生成し、依存関係分析を実行する
詳細は Dependency Analysis を参照のこと

device-buffer-size
デバイスバッファサイズ
CDP 以外の操作のプロファイリングデータを格納するために予約されているデバイスメモリサイズ(MB 単位)を、同時カーネルトレースのため、コンテキスト上の各バッファに設定する
サイズは正の整数であること

device-cdp-buffersize
コンテキストの各バッファの CDP 操作のプロファイリングデータを保存するために予約されているデバイスメモリサイズ(MB 単位)を設定する
サイズは正の整数であること

devices
デバイス
後続の-events、-metrics、-query-events、および--query-metrics オプションのスコープを変更す

る詳細は Profiling Scope を参照のこと

event-collection-mode
イベント収集モード
すべてのイベント/メトリックのイベント収集モードを選択する
*kernel: イベント/メトリックは、カーネル実行の期間のみ収集される
*continuous: イベント/メトリックは、アプリケーションの期間中に収集される
これは、Tesla 以外のデバイスには適用されない
このモードは、NVLink イベント/メトリックとのみ互換性がある
このモードは、-profileall-processes または--profilechild-processes または--replaymode kernel または--replay-mode アプリケーションと互換性がない

events (e)
イベント(e)
特定のデバイスでプロファイルするイベントを指定する
複数のイベント名をカンマで区切って指定する
プロファイルされるデバイスは、-devices オプションによって制御される
それ以外の場合、イベントはすべてのデバイスで収集される
使用可能なイベントのリストを表示すには、-query-events を使用する
--events all を使用して、各デバイスで使用可能なすべてのイベントをプロファイリングする
特定のカーネル呼び出しを選択するには、-devices および--kernels を使用する

kernel-latency-timestamps
カーネル待ち時間タイムスタンプ
カーネルのレイテンシタイムスタンプの収集のオン/オフ、キューと送信
キューに入れられたタイムスタンプは、カーネル起動コマンドが CPU コマンドバッファのキューに入れられたときにキャプチャされる
送信されたタイムスタンプは、このカーネル起動を含む CPU コマンドバッファが GPU に送信された日時を示す
このオプションをオンにすると、プロファイリング中にオーバーヘッドが発生する可能性がある

kernels
カーネル
後続の-events、-metrics オプションのスコープを変更する
構文は次のとおり
* {kernel name}:指定されたカーネル名にスコープを制限する
* {[context id/name]:[stream id/name]:[kernel name]:[invocation]:
コンテキスト/ストリーム ID、名前、カーネル名、および呼び出しは、正規表現にする

空の文字列は、任意の数字または文字と一致する
[context id/name] または stream id/name]が正の数の場合、CUDA コンテキスト/ストリーム ID と厳密に照合される
それ以外の場合は、正規表現として扱われ、NVTX ライブラリで指定されたコンテキスト/ストリーム名と照合される
呼び出し回数が正の数の場合、カーネルの呼び出しと厳密に一致する
それ以外の場合は、正規表現として扱われます
例:--kernels "1:foo:bar:2"は、名前に "bar"が含まれ、コンテキスト 1 およびストリーム "foo"の 2 番目のインスタンスであるカーネルをプロファイルする
詳細は Profiling Scope を参照のこと

metrics (m)
特定のデバイスでプロファイルするメトリックを指定する
コンマで区切られた複数のメトリック名を指定する
プロファイルされるデバイスは、-devices オプションによって制御される
それ以外の場合、メトリックはすべてのデバイスで収集される
使用可能なメトリックのリストは、-query-metrics を使用する
各デバイスで使用可能なすべてのメトリックをプロファイルするには、-metrics all を使用する
特定のカーネル呼び出しを選択するには、-devices および--kernels を使用する
注:-metrics all には、Visual Profiler のソースレベル分析に必要な一部のメトリックが含まれない
そのためには、-analysis-metrics を使用する

pc-sampling-period
pc サンプリング周期
サンプリングレコードがダンプされる PC サンプリング周期をサイクル単位で指定する
周期に使用する値は、5 から 31 の整数である
これにより、サンプリング周期が(2 ^ period)サイクルに設定される
注:GM20X+でのみ可能

profile-all-processes
この nvprof インスタンスを起動した同じユーザーが起動したすべてのプロセスをプロファイルする
注:このオプションで同時に実行するのは、nvprof の 1 つのインスタンスのみである
このモードでは、実行するアプリケーションを指定する必要はない
詳細は Multiprocess Profiling を参照のこと

profile-api-trace
プロファイル API トレース

CUDA ランタイム/ドライバ API トレースをオン/オフにする
* none:API トレースをオフにする
* runtime:CUDA ランタイム API トレースのみをオンにする
* driver:CUDA ドライバ API トレースのみをオンにする
* all:すべての API トレースをオンにする

profile-child-processes
プロファイル子プロセス
アプリケーションとそれによって起動されたすべての子プロセスをプロファイルする
詳細は Multiprocess Profiling を参照のこと

profile-from-start
プロファイル開始
アプリケーションの最初からプロファイリングを有効/無効にする
無効になっている場合、アプリケーションは{cu、cuda} Profiler {Start、Stop}を使用してプロファイリングをオンまたはオフにする
詳細は Focused Profiling を参照のこと

profiling-semaphore-pool-size
プロファイリングセマフォプールサイズ
各コンテキストのシリアル化されたカーネルとメモリ操作のプロファイリングデータを格納するために予約されているプロファイリングセマフォプールサイズを設定する
サイズは正の整数であること

query-events
クエリイベント
デバイスで利用可能なすべてのイベントを一覧表示す
照会されるデバイスは、-devices オプションで制御する

query-metrics
クエリ指標
デバイスで使用可能なすべてのメトリックを一覧表示す
照会されるデバイスは、-devices オプションで制御する

replay-mode
リプレイモード
すべてのイベント/メトリックを 1 回の実行で収集できない場合に使用される再生モードを選択する

*disabled:再生が無効になり、プロファイリングできなかったイベント/メトリックは削除される
*kernel:各カーネルの呼び出しが再生される
*application:アプリケーション全体が再生される
このモードは--profileall-processes または profilechild-processes と互換性がない

skip-kernel-replay-save-restore
このオプションを有効にすると、各カーネルパスの変更可能な状態の保存と復元がスキップされるため、カーネルの再生速度が大幅に向上する
入力/出力バッファの保存/復元をスキップすると、コンテキスト上のすべてのプロファイルされたカーネルが実行中に入力バッファの内容を変更しないように指定したり、デバイスヒープを残したままデバイス malloc/free または new/delete を呼び出したりする別の状態
具体的には、カーネルは同じ起動でバッファを malloc して解放するが、一致しない malloc または一致しない free を呼び出すことはできない
注:カーネルの 1 つが入力バッファを変更したり、一致しない malloc/free を使用しているときにこのモードを誤って使用すると、カーネルの実行エラーやデバイスデータの破損など、未定義の動作が発生する
* on:入力/出力バッファの保存/復元をスキップ
* off:各カーネル再生パスの入力/出力バッファを保存/復元

source-level-analysis (a)
ソースレベル分析(a)
特定のカーネル呼び出しでプロファイルされるソースレベルのメトリックを指定する
--devices および--kernels を使用して、特定のカーネル呼び出しを選択する
これらの 1 つ以上をコンマで区切って指定する
* global_access:グローバルアクセス
* shared_access:共有アクセス
* branch:分岐ブランチ
* instruction_execution:命令の実行
* pc_sampling:pc サンプリング、GM20X +でのみ使用可能
注:エクスポートファイルを指定するには、-export-profile を使用する
詳細は Source-Disassembly View を参照のこと

system-profiling
システムプロファイリング
電源、クロック、熱プロファイリングをオン/オフにする
詳細は System Profiling を参照のこと

timeout (t)

タイムアウト(t)
CUDA アプリケーションの実行タイムアウト(秒単位)を設定する
注:タイムアウトは、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
詳細は TimeoutFlush Profile Data のフラッシュを参照のこと

track-memory-allocations
トラックメモリ割り当て
タイムスタンプ、メモリサイズ、メモリタイプ、およびメモリ割り当てと解放のプログラムカウンタの記録を含む、メモリ操作の追跡をオン/オフにする
このオプションをオンにすると、プロファイリング中にオーバーヘッドが発生する可能性がある

unified-memory-profiling
統合メモリプロファイリング
プロセスデバイスごと、オフプロセスデバイスごと統合メモリプロファイリングを構成する
* per-process-device:各プロセスおよび各デバイスのカウントを収集する
* off:統合メモリプロファイリングをオフにする
詳細は「Unified Memory Profiling」を参照のこと

3.1.2. CPU プロファイリングオプション

cpu-profiling
CPU プロファイリング
CPU プロファイリングをオンにする
注:CPU プロファイリングは、マルチプロセスモードではサポートされない

cpu-profiling-explain-ccff
Common Compiler Feedback Format(CCFF)メッセージの解釈に使用する PGI pgexplain.xml ファイルへのパスを設定する

cpu-profiling-frequency
CPU プロファイリング周波数
CPU プロファイリング周波数を 1 秒あたりのサンプル数で設定する
最大は 500Hz である

cpu-profiling-maxdepth
各呼び出しスタックの最大深度を設定する

cpu-profiling-mode
CPU プロファイリングの出力モードを設定する
* flat:フラットプロファイルを表示
* top-down:親関数を上部に表示す
* bottom-up:親機能を下部に表示す

cpu-profiling-percentage-threshold
設定したパーセンテージしきい値を下回っているエントリを除外する
制限は 0 から 100 までの整数であること

cpu-profiling-scope
CPU プロファイリングスコープ
プロファイリングスコープを選択する
* function:スタックトレースの各レベルは個別の関数を表す
* instruction:スタックトレースの各レベルは、個別の命令アドレスを表す

cpu-profiling-show-ccff
バイナリに埋め込まれた Common Compiler Feedback Format(CCFF)メッセージを表示するかどうかを選択する
注:このオプションは--cpu-profilingscope 命令を意味する

cpu-profiling-show-library
各サンプルのライブラリ名を表示するかどうかを選択する

cpu-profiling-thread-mode
CPU プロファイリングのスレッドモードを設定する
* separate:スレッドごとに個別のプロファイルを表示
* aggregated:すべてのスレッドからのデータを集計

cpu-profiling-unwind-stack
各サンプルポイントで CPU コールスタックを戻すかどうかを選択する

openacc プロファイリング
OpenACC プロファイリングインターフェースからの情報の記録を有効/無効にする
注:OpenACC プロファイリングインターフェイスが使用するかどうかは、OpenACC ランタイムによって異なる
詳細は OpenACC を参照のこと

openmp プロファイリング
OpenMP プロファイリングインターフェースからの情報の記録を有効/無効にする
注:OpenMP プロファイリングインターフェースが使用するかどうかは、OpenMP ランタイムによって異なる
詳細は OpenMP を参照のこと

3.1.3 表示オプション

context-name
CUDA コンテキストの名前
コンテキスト名文字列の%i は、コンテキストの ID に置き換えられる
コンテキスト名文字列の%p は、プロファイリングされるアプリケーションのプロセス ID に置き換えられる
コンテキスト名文字列の%q {<ENV>}は、環境変数<ENV>の値に置き換えられる環境変数が設定されない場合エラーになる
コンテキスト名文字列の%h は、システムのホスト名に置き換えられる
コンテキスト名文字列の%%は%に置き換えられる
%に続く他の文字はすべて無効である

csv
出力にコンマ区切り値を使用する
詳細は CSV を参照のこと

demangling
デマングル
関数名の C ++名のデマングルをオン/オフにする
詳細は Demangling を参照のこと

normalized-time-unit (u)
正規化された時間単位(u)
出力で使用される時間の単位を指定する
s:秒
ms:ミリ秒
ns:ナノ秒
col:各列の固定単位
auto:スケールは、長さに基づいて各値に対して選択される
詳細は Adjust Units を参照のこと

openacc-summary-mode
OpenACC サマリーでの期間の計算方法を設定する
詳細は OpenACC Summary Modes を参照のこと

trace
トレースするオプション(またはコンマで区切られたオプション)を指定する
* api: CUDA ランタイムとドライバ API トレースのみをオンにする
* gpu: CUDA GPU トレースのみをオンにする

print-api-summary
CUDA ランタイム/ドライバ API 呼び出しの概要を出力する

print-api-trace
CUDA ランタイム/ドライバ API トレースを出力する
詳細は GPU-Trace and API-Trace Modes を参照のこと

print-dependency-analysis-trace
依存性分析トレース表示
依存関係分析トレースを出力する
詳細は Dependency Analysis を参照のこと

print-gpu-summary
GPU でのアクティビティの要約を出力する(CUDA カーネルと memcpy/memset を含む)
詳細は Summary Mode を参照のこと

print-gpu-trace
個々のカーネル呼び出し(CUDA memcpy/memset を含む)を出力し、時系列順に並べ替えます
イベント/メトリックプロファイリングモードでは、各カーネル呼び出しのイベント/メトリックを表示す
詳細は GPU-Trace and API-Trace Modes を参照のこと

print-openacc-constructs
OpenACC プロファイルに親構成名を含める
詳細は OpenACC オプションを参照のこと

print-openacc-summary
OpenACC プロファイルの概要を表示する

print-openacc-trace
OpenACC プロファイルのトレースを表示する

print-openmp-summary
OpenMP プロファイルの概要を表示する

print-summary (s)
プロファイリング結果の概要を画面に出力する
注:--export-profile または他の表示オプションが使用されない限り、これがデフォルトである

print-summary-per-gpu
各 GPU のプロファイリング結果の概要を出力する

process-name
プロセスの名前
プロセス名文字列の%p は、プロファイリングされるアプリケーションのプロセス ID に置き換えられる
プロセス名文字列の%q {<ENV>}は、環境変数<ENV>の値に置き換えられる環境変数が設定されない場合、エラーになる
プロセス名文字列の%h は、システムのホスト名に置き換えられる
プロセス名文字列の%%は%に置き換えられる
%に続く他の文字はすべて無効である

quiet
すべての nvprof 出力を抑制する

stream-name
ストリーム名
CUDA ストリームの名前
ストリーム名文字列の%i は、ストリームの ID に置き換えられる
ストリーム名文字列の%p は、プロファイリングされるアプリケーションのプロセス ID に置き換えられる
ストリーム名文字列の%q {<ENV>}は、環境変数<ENV>の値に置き換えられる環境変数が設定されない場合、エラーになる
ストリーム名文字列の%h は、システムのホスト名に置き換えられる
ストリーム名文字列の%%は%に置き換えられる %に続く他の文字はすべて無効である

3.1.4. IO オプション

export-profile (o)
エクスポートプロファイル(o)
後でインポートしたり、NVIDIA Visual Profiler で開いたりする結果ファイルをエクスポートする
ファイル名文字列の%p は、プロファイルされるアプリケーションのプロセス ID に置き換えられる
ファイル名文字列の%q {<ENV>}は、環境変数<ENV>の値に置き換えられる環境変数が設定されない場合、エラーになる
ファイル名文字列の%h は、システムのホスト名に置き換えられる
ファイル名文字列の%%は%に置き換えられる
%に続く他の文字はすべて無効である
デフォルトでは、このオプションは要約出力を無効にする
注:プロファイルされるアプリケーションが子プロセスを作成する場合、または--profile-allprocesses が使用される場合、各プロセスの正しいエクスポートファイルを取得するには%p 形式が必要である
詳細は Export/Import を参照のこと

force-overwrite (f)
強制上書き(f)
すべての出力ファイルを強制的に上書きする(既存のファイルは上書きされる)

import-profile (i)
前回の実行からの結果プロファイルをインポートする
詳細は Export/Import を参照のこと

log-file
ログファイル
nvprof にすべての出力を指定されたファイルまたは標準チャネルの 1 つに送信する
ファイルは上書きされる
ファイルが存在しない場合は、新しいファイルが作成される
%1 は、標準出力チャネル(stdout)を示す
%2 は、標準エラーチャネル(stderr)を示す
注:これはデフォルトである
%p は、プロファイルされるアプリケーションのプロセス ID に置き換えられる
%q {<ENV>}は、環境変数<ENV>の値に置き換えられる
環境変数が設定されない場合、エラーになる
%h は、システムのホスト名に置き換えられる

ファイル名の%%は%に置き換えられる
%に続く他の文字はすべて無効である詳細は出力のリダイレクトを参照のこと

print-nvlink-topology
nvlink トポロジーの表示

print-pci-topology
PCI トポロジを表示する

help (h)
ヘルプ情報を表示する

version (V)
このツールのバージョン情報を表示する

3.2 プロファイリングモード
nvprof は、以下に示すモードのいずれかで動作する

3.2.1 Summary Mode 要約モード
サマリーモードは、nvprof のデフォルトの動作モードである
このモードでは、nvprof は、アプリケーションによって実行される各カーネル関数および各タイプの CUDA メモリコピー/セットに対して単一の結果行を出力する
カーネルごとに、nvprof は、カーネルのすべてのインスタンスの合計時間またはメモリコピーのタイプと、平均、最小、および最大時間を出力する
カーネルの時間は、デバイスでのカーネル実行時間である
デフォルトでは、nvprof はすべての CUDA ランタイム/ドライバ API 呼び出しの要約も出力する
nvprof の出力(テーブルを除く)の前には== <pid> ==が付けられ、<pid>はプロファイルされるアプリケーションのプロセス ID である
以下は、CUDA サンプルの matrixMul で nvprof を実行する簡単な例である
$ nvprof matrixMul

※注
--profile-api-trace none を使用して、必要がない場合は API トレースをオフにするこれにより、特にカーネルが短い場合に、プロファイリングのオーバーヘッドの一部が削減される

複数の CUDA 対応デバイスがプロファイルされている場合、nvprof --print-summary-per-gpu を使用して、GPU ごとに 1 つの要約を出力する

nvprof は、サマリーモードで CUDA 動的並列処理をサポートする
アプリケーションが動的並列処理を使用する場合、出力には、ホスト起動カーネルの数とデバイス起動カーネルの数の 1 つの列が含まれる以下は、CUDA 動的並列処理のサンプル cdpSimpleQuicksort で nvprof を実行する例である

```
$ nvprof cdpSimpleQuicksort
```

3.2.2. GPU-Trace および API-Trace モード

GPU-Trace および API-Trace モードは個別にまたは一緒に有効にする
GPU トレースモードは、GPU で発生するすべてのアクティビティのタイムラインを時系列で提供する
各カーネル実行とメモリコピー/セットインスタンスが出力に表示される
カーネルまたはメモリのコピーごとに、カーネルパラメータ、共有メモリの使用状況、メモリ転送のスループットなどの詳細情報が表示される
カーネル名の後の角括弧内に示されている番号は、そのカーネルを起動した CUDA API に対応する
次に例を示す

```
$ nvprof --print-gpu-trace matrixMul
```

nvprof は、GPU トレースモードでの CUDA 動的並列処理をサポートする
ホストカーネルの起動では、カーネル ID が表示される
デバイスカーネルの起動では、カーネル ID、親カーネル ID、および親ブロックが表示される
次に例を示す

```
$ nvprof --print-gpu-trace cdpSimpleQuicksort
```

API トレースモードでは、ホスト上で呼び出されたすべての CUDA ランタイムとドライバ API 呼び出しの時系列が時系列で表示される次に例を示す

```
$ nvprof --print-api-trace matrixMul
```

プロファイラのセットアップ方法により、最初の「cuInit()」ドライバ API 呼び出しはトレースされない

3.2.3 Event/metric Summary Mode イベント/メトリックサマリーモード

特定の NVIDIA GPU で利用可能なすべてのイベントのリストを表示すには、-query-events オプションを使用する
特定の NVIDIA GPU で使用可能なすべてのメトリックのリストを表示すには、-query-metrics オプションを使用する
nvprof は、複数のイベント/メトリックを同時に収集する次に例を示す

```
$ nvprof --events warps_launched、local_load --metrics ipc matrixMul
```

指定されたイベント/メトリックをアプリケーションの 1 回の実行でプロファイルできない場合、nvprof はデフォルトで、すべてのイベント/メトリックが収集されるまで各カーネルを複数回再生する
--replay-mode <mode>オプションを使用して、再生モードを変更する
「application replay」モードでは、nvprof は、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のではなく、アプリケーション全体を再実行する
アプリケーションが大量のデバイスメモリを割り当てる場合、このモードはカーネル再生モードよりも高速になる場合がある
再生を完全にオフにするその場合、プロファイラは一部のイベント/メトリックを収集しない
各デバイスで利用可能なすべてのイベントを収集するには、オプション--events all を使用する
各デバイスで使用可能なすべてのメトリックを収集するには、オプション--metrics all を使用する

すべてのカーネル実行が GPU でシリアル化されるため、イベントまたはメトリックの収集により、アプリケーションの全体的な性能特性が大幅に変わる可能性がある
多数のイベントまたはメトリックが要求された場合、どの再生モードを選択しても、アプリケーション全体の実行時間が大幅に増加する可能性がある

3.2.4 イベント/メトリックトレースモード
イベント/メトリックトレースモードでは、カーネルの実行ごとにイベントとメトリックの値が表示される
デフォルトでは、イベントとメトリックの値は GPU のすべてのユニットで集計される
たとえば、マルチプロセッサ固有のイベントは、GPU 上のすべてのマルチプロセッサにわたって集計される
--aggregate-mode off を指定すると、各ユニットの値が表示される
たとえば、次の例では、GPU の各マルチプロセッサの「ブランチ」イベント値が表示される

```
$ nvprof --aggregate-mode off --events local_load --print-gpu-trace matrixMul
```

--aggregate-mode はメトリックに適用されるが、一部のメトリックは集約モードでのみ使用でき、一部は非集約モードでのみ使用する

3.3 プロファイリング制御
3.3.1 タイムアウト
nvprof にタイムアウト(秒単位)を提供する
プロファイルされる CUDA アプリケーションは、タイムアウト後に nvprof によって強制終了される
タイムアウト前に収集されたプロファイリング結果が表示される

タイムアウトは、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

3.3.2　Concurrent Kernels 並行カーネル
同時カーネルプロファイリングがサポートされ、デフォルトでオンににになっている
この機能をオフにするには、オプション--concurrent-kernels off を使用する
これにより、CUDA アプリケーションが nvprof で実行されるときに、カーネルの同時実行が強制的にシリアル化される

3.3.3 プロファイリングの範囲
イベント/メトリックを収集するとき、nvprof は、デフォルトであるべての表示可能な CUDA デバイスで起動されたすべてのカーネルをプロファイルする
このプロファイリングの範囲は、次のオプションによって制限する
--devices <デバイス ID>は、それに続く--events、-metrics、-query-events、および--query-metrics オプションに適用される
これらのオプションを制限して、<デバイス ID>で指定されたデバイスでのみイベント/メトリックを収集するこれは、コンマで区切られたデバイス ID 番号のリストにする
--kernels <kernel filter>は、それに続く--events および--metrics オプションに適用される
これらのオプションを制限して、<kernel filter>で指定されたカーネルでのみイベント/メトリックを収集する
これは以下の構文である
<kernel name>
または
<context id/name>:<stream id/name>:<kernel name>:<invocation>
山括弧内の各文字列は、標準の Perl 正規表現にする
空の文字列は、任意の数字または文字の組み合わせと一致する
呼び出し番号 n は、カーネルの n 番目の呼び出しを示す
呼び出しが正の数の場合、カーネルの呼び出しと厳密に一致する
それ以外の場合は、正規表現として扱われます
呼び出し番号は、カーネルごとに個別にカウントされる
したがって、たとえば::: 3 は、すべてのカーネルの 3 番目の呼び出しに一致する

コンテキスト/ストリーム文字列が正の数の場合、cuda コンテキスト/ストリーム ID と厳密に照合される
それ以外の場合は、正規表現として扱われ、NVIDIA Tools Extension によって提供されるコンテキスト/ストリーム名と照合される

--devices と--kernels の両方を複数回指定でき、個別のイベント/メトリックが関連付けられる

--events、-metrics、-query-events、および--query-metrics は、それらの前の最も近いスコープオプションによって制御される
例として、次のコマンド、
　nvprof --devices 0 --metrics ipc --kernels "1:foo:bar:2" --events local_load a.out
デバイス 0 で起動されたすべてのカーネルのメトリック ipc を収集する
また、名前に bar が含まれ、コンテキスト 1 およびデバイス 0 のストリーム foo で起動された 2 番目のインスタンスであるカーネルのイベント local_load も収集する

3.3.4 マルチプロセスプロファイリング
デフォルトでは、nvprof はコマンドライン引数で指定されたアプリケーションのみをプロファイルする
そのプロセスによって起動された子プロセスは追跡しない
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には、-profile-child-processes オプションを使用する

nvprof は、fork()を実行するプロセスをプロファイルできないが、exec()を実行しない

nvprof には、「全プロセスのプロファイル」モードもあるこのモードでは、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
「Ctrl-c」を入力してこのモードを終了する
CPU プロファイリングは、マルチプロセスモードではサポートされない

3.3.5 システムのプロファイリング
システムプロファイリングをサポートするデバイスの場合、nvprof は、アプリケーションが使用する各 GPU の電力、クロック、および熱的動作の低周波サンプリングを有効にする
この機能はデフォルトでオフににになっている
この機能をオンにするには、-system-profiling on を使用する
各サンプルポイントの詳細を表示すには、上記のオプションを--print-gputrace と組み合わせる

3.3.6 ユニファイドメモリプロファイリング
ユニファイドメモリをサポートする GPU の場合、nvprof は、システム上の各 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を収集する
この機能はデフォルトでオフににになっている
この機能をオンにするには、-unified-memory-profiling <per-processdevice | off>を使用する
各メモリ転送の詳細を表示すには、上記のオプションを--print-gpu-trace と組み合わせる
ユニファイドメモリをサポートするデバイスのペア間の P2P サポートのないマルチ GPU 構成では、管理されたメモリ割り当てはゼロコピーメモリに配置される
この場合、ユニファイドメモリプロファイリングはサポートされない

場合によっては、環境変数 CUDA_MANAGED_FORCE_DEVICE_ALLOC を設定して、管理された割り当てを強制的にデバイスメモリに配置し、これらのハードウェア構成での移行を有効にする
この場合、ユニファイドメモリプロファイリングがサポートされる
通常、CUDA が P2P をサポートする GPU のみを使用するように制限するには、環境変数 CUDA_VISIBLE_DEVICES を使用することを推奨する
詳細は CUDA C ++プログラミングガイドの環境変数の章を参照のこと

3.3.7. CPU スレッドトレース
正しい依存関係分析を可能にするために、nvprof は CPU 側のスレッド API に関する情報を収集する
これは、測定中に--cpu-thread-tracing on を指定することで有効にする
この情報を記録する必要がある場合
-アプリケーションが複数の CPU スレッドを使用し、
-これらのスレッドの少なくとも 2 つが CUDA API を呼び出する

現在、POSIX スレッド(Pthreads)のみがサポートされる
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る場合がある
nvprof は、実行動作のモデル化に必要な呼び出しを検出し、他の呼び出しをフィルタリングしようとする
フィルタ処理された呼び出しには、pthread_mutex_lock と pthread_mutex_unlock が含まれるこれらの呼び出しによって並行スレッドがブロックされない場合である

CPU スレッドのトレースは、Windows では使用できない

CPU スレッドのトレースは、最初の CUDA API 呼び出しの後に、この呼び出しを発行するスレッドから開始される
したがって、アプリケーションは、たとえば CUDA API を呼び出す他のユーザースレッドを生成する前に、メインスレッドから cuInit を呼び出する

3.4 出力
3.4.1 単位の調整デフォルトでは、nvprof は時間単位を自動的に調整して、最も正確な時間値を取得する
--normalized-time-unit オプションを使用して、結果全体で固定時間単位を取得する

3.4.2. CSV
プロファイリングモードごとに、オプション--csv を使用して、コンマ区切り値(CSV)形式で出力を生成する

結果は、Excel などのスプレッドシートソフトウェアに直接インポートする

3.4.3Export/Import エクスポート/インポート
各プロファイリングモードについて、オプション--export-profile を使用して結果ファイルを生成する
このファイルは人間が読める形式ではないが、オプション--import-profile を使用して nvprof にインポートしたり、Visual Profiler にインポートしたりする

プロファイラは、エクスポートプロファイルの形式として SQLite を使用する
このような形式でファイルを書き込むには、プレーンファイルを書き込むよりも多くのディスク操作が必要になる場合がある
したがって、ネットワークドライブなどの低速のデバイスにプロファイルをエクスポートすると、アプリケーションの実行が遅くなる可能性がある

3.4.4 Demangling デマングル
デフォルトでは、nvprof は C ++関数名をデマングルする
この機能をオフにするには、オプション--demangling off を使用する

3.4.5 Redirecting Output 出力のリダイレクト
デフォルトでは、nvprof はほとんどの出力を stderr に送信する
出力をリダイレクトするには、-log-file を使用する
--log-file%1 は、すべての出力を stdout にリダイレクトするよう nvprof に指示す
--logfile <filename>は、出力をファイルにリダイレクトする
ファイル名に%p を使用して、nvprof のプロセス ID、%h をホスト名、%q{ENV}を環境変数 ENV の値、%%を%で置き換えます

3.4.6 Dependency Analysis 依存関係分析
nvprof は、--dependency-analysis オプションを使用して、アプリケーションのプロファイルが作成された後に依存関係分析を実行する
この分析は、インポートされたプロファイルにも適用する
測定中に完全な CUDA API および GPU アクティビティトレースを収集する必要がある
--profile-apitrace none を使用して無効にしない場合、これは nvprof のデフォルトである

複数の CPU スレッドから CUDA を使用するアプリケーションの場合、CPU スレッドトレースも有効にする必要がある
オプション--print-dependency-analysis-trace を指定して、サマリー出力からトレース出力に変更し、関数タイプごとではなく関数インスタンスごとのクリティカルパスの時間などの計算されたメトリックを表示する

関数タイプごとに集計されたすべての計算されたメトリックを含む依存関係分析の要約出力の例を以下に示す
テーブルは、最初にクリティカルパスの時間でソートされ、次に待機時間でソートされる
概要には、その他の名前のエントリが含まれ、nvprof によって追跡されないすべての CPU アクティビティ(アプリケーションのメイン関数など)を参照する

3.5. CPU サンプリング

ボトルネックをよりよく理解し、CUDA アプリケーション全体の潜在的なホットスポットを特定するために、アプリケーションの CPU 部分のプロファイルを作成すると役立つ場合がある
アプリケーションの CPU 部分では、nvprof はプログラムカウンターをサンプリングし、特定の頻度でスタックを呼び出す
次に、データを使用してグラフを作成するノードは各呼び出しスタックのフレームである
可能であれば、関数とライブラリのシンボルも抽出される
グラフの例を以下に示す

グラフはさまざまな「ビュー」(トップダウン、ボトムアップ、またはフラット)で表示するため、ユーザーはさまざまな視点からサンプリングデータを分析する
たとえば、ボトムアップビュー(上記を参照)は、アプリケーションがほとんどの時間を費やしている「ホット」機能を識別に役立つ
トップダウンビューでは、メイン関数から始めて、アプリケーション実行時間の内訳が表示され、頻繁に実行される「呼び出しパス」を見つける
デフォルトでは、CPU サンプリング機能は無効ににになっている
これを有効にするには、オプション--cpuprofiling on を使用する
次の章では、CPU サンプリング動作を制御するすべてのオプションについて説明する

CPU サンプリングは、Intel x86/x86_64 アーキテクチャー用の Linux、Mac OS および Windows、および IBM POWER アーキテクチャーでサポートされる

POSIX システムで CPU プロファイリング機能を使用する場合、プロファイラは定期的な信号を送信してアプリケーションをサンプリングする
したがって、アプリケーションは、システムコールが中断されたときに適切に処理されるようにする必要がある

Windows では、nvprof はシンボル情報を解決するために Visual Studio のインストール(2010 以降)およびコンパイラが生成した.PDB(プログラムデータベース)ファイルを必要とする
アプリケーションをビルドするときは、.PDB ファイルが作成され、プロファイルされた実行可能

ファイルとライブラリの横に配置されていることを確認する
s
3.5.1CPU サンプリングの制限
以下は、現在のリリースの既知の問題である
-モバイルデバイスでは CPU サンプリングはサポートされない
-CPU サンプリングは現在、マルチプロセスプロファイリングモードではサポートされない
-一部のコンパイラの最適化、特にフレームポインタの省略と関数のインライン展開では、結果のスタックトレースが完全でない場合がある
-CPU サンプリング結果は CSV モードをサポートしていません
-Mac OSX では、まれにプロファイラがハングすることがある

3.6. OpenACC
64 ビット Linux プラットフォームでは、nvprof は CUPTI アクティビティ API を使用した OpenACC アクティビティの記録をサポートする
これにより、基になるコンパイラ生成の CUDA API 呼び出しに加えて、OpenACC 構成要素のレベルで性能を調査する
nvprof での OpenACC プロファイリングでは、ターゲットアプリケーションが PGI OpenACC ランタイム 19.1 以降を使用する必要がある
OpenACC アクティビティの記録は x86_64 および IBM POWER Linux システムでのみサポートされるが、以前に生成されたプロファイルデータのインポートおよび表示は、nvprof でサポートされるすべてのプラットフォームで利用する
OpenACC の要約出力の例を以下に示す
CUPTI OpenACC アクティビティは、ソースファイルと行情報を使用して、元の OpenACC 構成にマップされる
acc_enqueue_launch アクティビティーの場合、OpenACC コンパイラによって生成された起動済みの CUDA カーネル名がさらに表示される
デフォルトでは、nvprof は OpenACC コンパイラによって生成されたカーネル名を復号化する --demangling off を渡して、この動作を無効にする

3.6.1. OpenACC オプション
表 1 に、nvprof の OpenACC プロファイリング関連のコマンドラインオプションを示す
表 1 OpenACC オプション

--openacc-profiling <on | off>
OpenACC プロファイリングをオン/オフにする
注:OpenACC プロファイリングは、x86_64 および IBM POWER Linux でのみサポートされるデフォルトはオンである

--print-openacc-summary
記録されたすべての OpenACC アクティビティの概要を表示する

--print-openacc-trace
各アクティビティのタイムスタンプと期間を含む、記録されたすべての OpenACC アクティビティの詳細なトレースを表示する

--print-openacc-constructs
OpenACC アクティビティが発行される原因となった OpenACC 親構成の名前を含めます
19.1 より前の PGI OpenACC ランタイムを使用するアプリケーションの場合、この値は常に不明であることに注意する

--openacc-summary-mode <exclusive | inclusive>
OpenACC サマリーでのアクティビティ期間の表示方法を指定する
許可される値:
exclusive-排他的な期間(デフォルト)
inclusive-包括的期間
詳細は OpenACC サマリーモードを参照のこと

3.6.2 OpenACC サマリーモード
nvprof は、OpenACC サマリーモードで OpenACC アクティビティ期間を表示すための 2 つのモード(--print-openacc-summary で有効化)をサポートする:「排他的」と「包括的」
* Inclusive:
このモードでは、すべての期間がアクティビティの実際の実行時間を表す
これには、このアクティビティおよびそのすべての子(呼び出し先)で費やされた時間が含まれる
* Exclusive:
このモードでは、すべての期間がこのアクティビティのみで費やされた時間を表す
これには、このアクティビティに費やされた時間が含まれるが、そのすべての子(呼び出し先)の実行時間は含まれません

例として、それ自体が acc_enqueue_launch を呼び出してデバイスにカーネルを起動する OpenACC acc_compute_construct と、このカーネルの完了を待機する acc_implicit_wait を考えます
Inclusive モードでは、acc_compute_construct の期間には、acc_enqueue_launch と acc_implicit_wait で費やされた時間が含まれる
Exclusive モードでは、これら 2 つの期間が差し引かれます
要約プロファイルでは、これは、長い acc_compute_construct が高い起動オーバーヘッドを表しているのか、それとも長い待機(同期)時間を表しているのかを識別に役立つ

3.7. OpenMP

64 ビット Linux プラットフォームでは、nvprof は OpenMP アクティビティの記録をサポートする nvprof での OpenMP プロファイリングでは、ターゲットアプリケーションが OpenMP ツールインターフェイス(OMPT)をサポートするランタイムを使用する必要がある
(LLVM コードジェネレーターを使用する PGI バージョン 19.1 以降は OMPT をサポートする)
OpenMP アクティビティの記録は x86_64 および IBM POWER Linux システムでのみサポートされるが、以前に生成されたプロファイルデータのインポートおよび表示は、nvprof でサポートされるすべてのプラットフォームで利用する
OpenMP サマリー出力の例を以下に示す

3.7.1. OpenMP オプション

表 2 に、nvprof の OpenMP プロファイリング関連のコマンドラインオプションを示す

表 2 OpenMP オプション

--print-openmp-summary
記録されたすべての OpenMP アクティビティの概要を表示する

訳 佐賀大学 今井康貴 imaiy@cc.saga-u.ac.jp