第 8 章 Dependency Analysis 依存関係の分析

依存関係分析機能により、プログラムのランタイムの最適化と、複数の CPU スレッドと CUDA ストリームを利用するアプリケーションの同時実行性が可能になる
特定の実行のクリティカルパスを計算し、待機時間を検出し、異なるスレッドまたはストリームで実行されている関数間の依存関係を検査する

8.1 背景

nvprof および Visual Profiler の依存関係分析は、アプリケーションの実行トレースに基づく
トレースは、タイムスタンプと期間とともに、API 関数呼び出しや CUDA カーネルなどのすべての関連アクティビティをキャプチャする
この実行トレースと、異なるスレッド/ストリームでのこれらのアクティビティ間の依存関係のモデルが与えられると、依存関係グラフを構築できる
このグラフでモデル化された一般的な依存関係は、CUDA カーネルがその起動 API コール前に開始できなくさせる
またはこのストリームで以前にキューされたすべての作業が完了する前にブロッキング CUDA ストリーム同期コールを戻さない
これらの依存関係は、CUDA API コントラクトによって定義される
この依存関係グラフと API モデルから、待機状態を計算する
待機状態は、API 関数呼び出しなどのアクティビティが、別のスレッドまたはストリームのイベントを待機してブロックされている期間である
前のストリーム同期の例では、同期する API 呼び出しは、それぞれの CUDA ストリームで GPU アクティビティを待機する必要がある間ブロックされる
待機状態が発生する場所と機能がブロックされる期間についての知識は、アプリケーションでのより高いレベルの同時実行の最適化の機会を特定に役立つ
個々の待機状態に加えて、キャプチャされたイベントグラフのクリティカルパスにより、アプリケーションランタイム全体の原因となっている関数呼び出し、カーネル、メモリコピーを特定する
クリティカルパスは、待機状態を含まないイベントグラフを通る最長のパスであるつまり、このパスのアクティビティを最適化すると、実行時間を直接改善する

8.2Metrics 指標

Waiting Time
待機状態は、API 関数呼び出しなどのアクティビティが、別のスレッドまたはストリームのイベントを待機してブロックされている期間である
待機時間は、実行ストリーム間の負荷の不均衡を示す指標である
以下の例では、ブロッキング CUDA 同期 API 呼び出しは、それぞれのカーネルが GPU での実行を完了するのを待機する
すぐに待機するのではなく、カーネルの実行と同様のランタイムの同時 CPU 作業をオーバーラ

ップさせて、コンピューティングデバイス（CPU または GPU）がブロックされる時間を短縮する必要がある
（図）

Time on Critical Path
クリティカルパスの時間
クリティカルパスは、待機状態を含まないイベントグラフを通る最長のパスであるつまり、このパスのアクティビティを最適化すると、実行時間を直接改善する
クリティカルパスの時間が長いアクティビティは、アプリケーションの実行時間に大きな影響を与えます
以下の図の例では、CPU が cudeDeviceSynchronize で完了するのを待ってブロックされているため、copy_kernel はクリティカルパス上にある
カーネルランタイムを減らすと、CPU は API 呼び出しから早く戻り、プログラムの実行を続行する
一方、jacobi_kernel は CPU の作業と完全にオーバーラップするつまり、カーネルがすでに終了した後で、同期 API 呼び出しがトリガされる
このカーネルの終了を待機している実行ストリームはないため、その継続時間を短縮しても、全体的なアプリケーションランタイムは改善されない可能性がある

8.3 サポート
次のプログラミング API は、依存関係分析で現在サポートされる
* CUDA ランタイムとドライバ API
* POSIX スレッド（Pthreads）、POSIXmutexes、条件変数

依存関係分析は Visual Profiler と nvprof で利用する
依存関係分析ステージは、ガイドなしアプリケーション分析で選択でき、新しい依存関係分析コントロールがタイムラインで使用する nvprof でこの機能を使用する方法は、依存関係分析の章を参照のこと

8.4 制限事項
異なるスレッドと CUDA ストリーム間の依存関係と待機時間の分析では、サポートされているそれぞれの API コントラクトに記述されている実行依存関係のみが考慮される
これには特に、リソースの競合の結果としての同期は含まれません
たとえば、独立した CUDA ストリームにキューされた非同期メモリコピーは、具象 GPU にコピーエンジンが 1 つしかない場合でも、依存としてマークされない
さらに、分析では、サポートされない API を使用した同期は考慮されない
たとえば、あるメモリ位置の値をアクティブにポーリングしている（ビジー待機）CPU スレッドは、別の同時アクティビティではブロックされているとは見なされない

依存関係分析は、CUDA Dynamic Parallelism（CDP）を使用するアプリケーションのサポートを制限する
CDP カーネルは、CUPTI アクティビティ API を介して追跡されない GPU からの CUDA API 呼び出しを使用する
したがって、分析では、CDP カーネルの完全な依存関係と待機時間を特定できない
ただし、CDP カーネル間の親子起動の依存関係を利用する
その結果、クリティカルパスには常に、各ホスト起動カーネルの最後の CDP カーネルが含まれる

POSIX セマフォ API は現在サポートされない

依存関係分析は API 関数をサポートしていません
cudaLaunchCooperativeKernelMultiDevice または
cuLaunchCooperativeKernelMultiDevice
これらの API 関数のいずれかによって起動されたカーネルは、正しく追跡されない可能性がある

訳 佐賀大学 今井康貴 imaiy@cc.saga-u.ac.jp