2010 年 4 月 2 日 FS 専攻 修士論文中間発表会 発表資料

GPGPU プログラム開発におけるプロファイリング方式に関する研究

高性能コンピューティング学講座 本多・近藤研究室
0953009 佐々木 信
主任指導教員： 本多 弘樹

1 研究概要

近年 GPU を汎用計算に用いる GPGPU(General-purpose computing on graphics processing units) が注目されている。GPU は、CPU に比べ浮動小数点演算性能が高く、GPGPU では大規模並列計算において、CPU のみで計算する場合よりも数十倍〜数百倍の演算性能を発揮できる。

しかし、プログラムの最適化を行わないことには CPU のみで計算を行った場合よりも処理が遅くなってしまう可能性がある。GPU の性能を十分に発揮させるためには、GPU のアーキテクチャを考慮しながらプログラムの最適化を行う必要がある。しかし、最適化では、「コードの変更」と「プロファイリングによる性能測定」を交互に繰り返すなど多くの時間が掛かる。

そこで本研究では、「プロファイリングによる性能測定」の部分を効率的にし、最適化にかかる時間の短縮を図る。最適化の際は、変更していない部分の実行性能は変化しないのでプロファイリングを行う必要がない。故に本研究では、最適化のために変更した関数のみをプロファイリングし、それ以外の部分は実行しないことで、最適化の作業をより効率的に行えるようにする。

これにより、GPGPU プログラム開発者自身はプロファイリングのためにソースコードに手を入れることなしに、最適化を行った関数のみのプロファイルを得ることができるようになる。

2 CUDA

本研究では CUDA プログラミングモデルを用いる。

CUDA(Compute Unified Device Architecture) は、C 言語を拡張したプログラミングモデルで、従来よりも簡単に GPGPU を行うことができる [1]。CUDA では、GPU 側で処理したい部分をカーネル関数として記述し、CPU 側からこのカーネル関数を呼び出すという形で GPU を用いる。また、GPU からは CPU 側のメモリを直接操作できないので、カーネル関数を呼び出す前に、GPU で用いるデータを GPU 側へ転送する必要がある。

本研究では、このデータ転送に着目し、GPU 側に転送されるデータのみをログに保存することで、少ないオーバヘッドでプロファイリングの効率化ができる。

3 提案手法

CUDA プログラムの最適化を行う際、最適化の対象はカーネル関数とし、プロファイリングは複数回行うことを前提とする。CPU ルーチン内のカーネル関数呼び出しの前後にコードを自動的に挿入することで、カーネル関数を実行するのに必要なパラメータをログに保存する。２回目以降のプロファイリングの際、ログ内のパラメータを代入するコードをカーネル関数呼び出しの前に挿入する。２回目以降は変更したカーネル関数のみプロファイリングを行う。これにより、プロファイリングを高速化し、最適化を効率的に行えるようにする。

提案手法におけるプロファイリングの流れを以下に示す。(図 1)

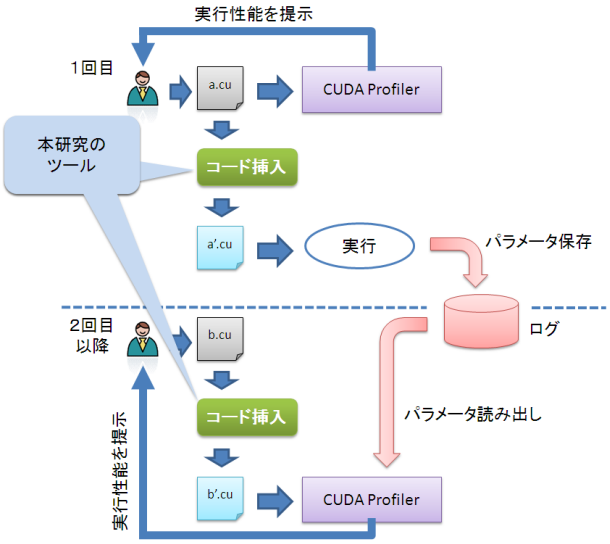

図 1: 提案手法

- 1 回目のプロファイリング時
  1. ＣＵＤＡプロファイラを起動し、プログラムの性能を測定。
  2. カーネル関数呼び出しの前後に、ログに保存するためのコードを挿入。(図 2)
  3. コンパイルする。
  4. プログラムを実行し、パラメータをログに保存。また、カーネル関数の結果をログに保存。

- 2 回目以降のプロファイリング時

  1. 変更したカーネル関数呼び出しの前に、ログから値を読みだすコードを挿入。カーネル関数呼び出しの後ろに、カーネル関数の結果をログに保存するためのコードを挿入。(図 3)
  2. 変更したカーネル関数の直前まで処理をスキップする。
  3. 変更したカーネル関数の直後で処理を終了させる。
  4. コンパイルする。
  5. ＣＵＤＡプロファイラを起動し、変更したカーネル関数の性能を測定。

```
int main(int argc, char** argv)
{
                :
        cudaMemcpy(d_A, h_A, mem_size_A,
                                cudaMemcpyHostToDevice);
        save_valuetolog(h_A, size_A, "inputA.dat");

         kernel<<< grid, threads >>>(d_B, d_A); //カーネル関数呼び出し

        save_valuetolog(h_B, size_B, "outputB1.dat");

                :
                :
}
                    :1回目プロファイリング時に挿入するコード
```

図 2: 1 回目プロファイリング時のコード挿入位置

```
int main(int argc, char** argv)
{
goto KERNEL;

                :
        cudaMemcpy(d_A, h_A, mem_size_A,
                                cudaMemcpyHostToDevice);
KERNEL:

        write_logtovalue(h_A, size_A, "inputA.dat");

        kernel<<< grid, threads >>>(d_B, d_A);  //カーネル関数呼び出し

        save_valuetolog(h_B, size_B, "outputB2.dat");

exit(0);
                :
                :
}
                    :2回目以降のプロファイリング時に挿入するコード
```

図 3: 2 回目以降プロファイリング時のコード挿入位置

なお、ログに保存するパラメータを以下に示す。

- CUP から GPU へ転送された変数の値。
- GPU 側のメモリ領域確保のためのサイズ。
- カーネル関数呼び出しの引数。
- カーネル関数呼び出しの際のパラメータ。(スレッド数、ブロック数)

4 関連研究

プロファイラとしては、CUDA と共に配布されている CUDA Visual Profiler が知られている [2]。これを用いることにより、ユーザは詳細なプロファイルを得ることができる。本研究でもこのプロファイラを用い実行性能の測定を行う。

また、ログへの保存を利用している支援ツールでは、滝沢ら [3] が、CUDA プログラムにおいてデバッギングの際、チェックポイントを設置し、データをログに保存することでその位置からのリスタートを可能にした。しかし、このチェックポイントは手動で設置する必要がある。

Qiming ら [4] は、デバッギングの際、GPU 内のデータフローを自動的にログに保存することで、原因の特定の難しいバグの発見を支援している。

これらの研究 [3][4] では、データをログに保存することで、デバッギングの支援を可能としている。本研究では、この手法をプロファイリングに用いる。

5 進捗

現在は、ログへの保存や、ログからの読み出しを行うコードを実際の CUDA プログラムに手動で挿入し、正しく動作するか確認中である。

6 今後の方針

自動的にコードを挿入するなどの機能を実装していく。提案手法では、CPU から呼び出されたカーネル関数を変更した場合、そのカーネル関数のみプロファイリングすることが可能である。しかし、CUDA では GPU からもカーネル関数の呼び出しが可能である。提案した手法では、GPU から呼び出されたカーネル関数のみのプロファイリングは行えない。それゆえ、この問題をどのように対処するかを検討する必要がある。

参考文献

[1] NVIDIA Corporation: CUDA Zone, http://www.nvidia.com/object/cuda_home_new.html

[2] NVIDIA Corporation: CUDA Visual Profiler, http://developer.nvidia.com/object/cuda_3_0_downloads.html

[3] 滝沢寛之, 佐藤功人, 小松一彦, 小林広明: CUDA アプリケーション向けチェックポイント・リスタート機能の実装と評価, 情報処理学会研究報告: Vol.2009-HPC-122 No.7 pp.1-7 (2009)

[4] Qiming Hou, Kun Zhou, Baining Guo: Debugging GPU Stream Programs Through Automatic Dataflow Recording and Visualization,*ACM Transactions on Graphics(SIGGRAPH Asia 2009), pp1-12* (2009)