2011 年 4 月 4 日 FS 専攻修士論文中間発表会発表資料

複数 GPU 向けの CUDA コードを生成する OpenMP 処理系の研究

高性能コンピューティング学講座 本多・近藤研究室
1053018 長塚　郁
主任指導教員：本多弘樹

1　研究概要

高い並列性と広いメモリバンド幅を持つ GPU(Graphics Processing Unit) が高性能コンピューティング (HPC) 分野で注目され，また統合開発環境 CUDA(C ompute Unified Device Architecture) の公開により，GPU を汎用計算に用いる GPGPU(General-Purpose computing on GPUs) が様々な分野で行われるようになった．更に，GPU を複数同時に用いることによって大規模計算を可能とし，さらなるアプリケーションの高速化を目指す研究もおこなわれている [1].

しかし，CUDA のプログラミングには特有の実行モデルやメモリモデルの知識を得る必要があり，OpenMP などの CPU 並列プログラミングを行っていたプログラマにとって負担が大きい．また CUDA は GPU デバイスなどの環境が整っていなければ使用することができないため，環境によっては大きくプログラムを改造する必要が出てくる．

また複数 GPU 間の直接通信や並列実行を行うために，CPU 側の並列プログラミングを行う手法があるが，並列処理が多層化することから更にプログラミングが複雑化する．

そこで本研究では簡易な並列プログラミングが可能な OpenMP に注目した．OpenMP 処理系コンパイラによって複数 GPU 向けのコード生成を可能にすることで，複数 GPU プログラミングの簡易化を図る．それにより，複数 GPU を用いた並列コンピューティングを行うプログラマの負担の軽減が期待できる．

2　研究背景

2.1　CUDA と単一 GPU プログラミング

CUDA は NVIDIA 社が公開した C(C++) 言語ベースの統合開発環境である．CUDA では並列計算を行う計算単位を GPU スレッドと呼び，この GPU スレッドが多数生成されることによって並列計算がおこなわれる．また GPU スレッドの集まりをブロック，ブロックの集まりをグリッドとして階層的に管理が行われ，同じグリッド内の GPU スレッドは同一の処理を行う．この際，その処理の内容はカーネル関数としてコードに書かれる．全 GPU スレッドは大容量低速メモリのデバイスメモリを共有してデータを参照することができる．そのため，単一 GPU での処理は共有メモリ型並列計算のように振る舞う．

2.2　複数 GPU プログラミング

計算の高速化において HPC 分野では複数 GPU を用いた研究が盛んに行われている [1][2].

CUDA で複数 GPU の並列実行を行う手法の一つに，マルチスレッドなどの API を用いてホスト側 (CPU) で並列プログラミングを行う方法がある．この場合，単一 GPU と比べて並列階層が一つ増える事になり，プログラムが複雑化しやすい問題がある．また，各 GPU が持つデバイスメモリはその GPU 自身のみが参照できる．そのため GPU 間でデータ参照が必要な場合は CPU に用意したバッファメモリを介する必要がある [2]．つまり，複数 GPU を用いる際は分散メモリ型並列計算のように振る舞う．

2.3　OpenMP

OpenMP はマルチスレッド並列プログラミングの API の一つであり，, 簡易的に並列プログラミングを行うことができる．実行モデルに fork-join 型並列モデルを採用している．プログラムで並列構文を発見した場合，複数 CPU スレッドが生成され，並列実行が行われる (fork)．各 CPU スレッドは並列構文内に書かれた処理を終了すると，他の CPU スレッドが終了するまで待ち，終了した後，並列処理をしていた CPU スレッドが解放され 1CPU スレッドで逐次処理される (join)．この際，並列実行の階層構造は 1 つとなり，CUDA GPU に比べて階層構造は少ないことが分かる．また，全て CPU スレッドはメインメモリを共有する．

3　提案手法

3.1　並列実行単位の対応付け

OpenMP と単一 GPU では，共有メモリ型である共通点を持ち，OpenMP の fork-join 型並列モデルは CUDA GPU におけるホスト側とデバイス側 (GPU) に置き換えることができる．しかし，複数 GPU の場合では分散メモリ型となるため，このギャップを解消する必要があるが，

本研究では CPU を GPU の共有メモリとして扱うことで解消する．

また，各実行形式の並列実行階層は異なる．forall ループの並列実行を行った場合，GPU 無しでは複数 CPU スレッドの並列階層 1 つで並列化が行われる．1GPU の場合，1CPU スレッドから起動した 1GPU の複数ブロックと複数 GPU スレッドの並列階層 2 つで並列化が行われる．複数 GPU の場合，複数 CPU スレッドでまず並列化され，それが更にそれぞれの GPU で複数ブロック，複数 GPU スレッドで並列化が行われるため，計 3 層の並列階層がある．本研究ではこれらの対応化を行っていく (Fig. 1 および Fig. 2)．

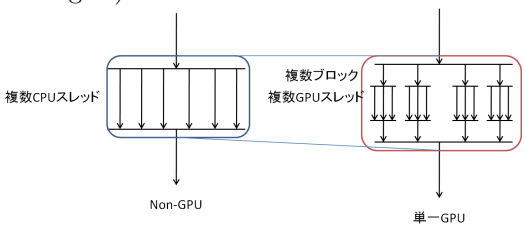

Fig. 1: OpenMP の並列実行階層の単一 GPU の対応付け．

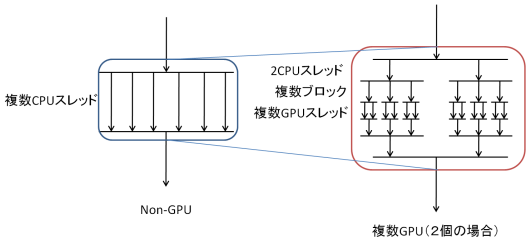

Fig. 2: OpenMP の並列実行階層の複数 GPU の対応付け．

3.2 メモリのマッピング

OpenMP のメモリモデルと GPU のメモリモデルを対応付ける必要がある．OpenMP のスレッド間シェアード変数に関して複数 GPU ではホスト側のバッファメモリにマッピングし，各 GPU が処理するのに必要な分だけデバイスメモリにコピーを行う事とする．この際，定数に関しては，書き込み専用のコンスタントメモリやテクスチャメモリにマッピングする．スレッドプライベート変数に指定されている変数は GPU のレジスタにマッピングする．

4 関連研究

単一 GPU を対象に CUDA コードを生成する OpenMP 処理系として OMPCUDA が提案されている [3]．これは OpenMP 処理系コンパイラ Omni OpenMP Compiler を基に作られたコンパイラで，中間コードに変換された OpenMP コードのうち並列処理を GPU 実行コードに，逐次処理を CPU 実行コードに変換する．本研究ではこのコンパイラを基に，複数 GPU 対応の OpenMP 処理系の制作を目指す．

また単一 GPU を対象にしたコード変換コンパイラとして，OpenMPC が提案されている [4]．これはコード変換コンパイラ Cetus Compiler を基に作られたコンパイラであり，自動最適化機構を持つが，複数 GPU には対応していない．

また複数 GPU プログラミングを行った研究事例として，Shimokawabe らの大規模な天気モデルの計算を行った例 [1] や小川らによる相変態を対象にした GPU の数に対する性能の拡張を測った例 [2] があげられる．これらの研究では同ノード内の複数 GPU データ通信を OpenMP で，異なるノード同士の複数 GPU データ通信を MPI を用いて行われている．本研究では異なるノード間通信が必要なものは対象としない．

5 進捗

GPU を取り扱うに当たって，その知識を得るために CUDA アーキテクチャの理解や複数 GPU プログラミングの試行を行った．また OMPCUDA の複数 GPU プログラミングへの対応を進めている．複数 GPU プログラミングの試行においては，forloop による行列和と行列差計算を試行し，実行時間の比較などを行った．また GPU 間のデータ通信の特性やメモリの生成と解放の実験を行い，それぞれ実行時間を測定した．

6 今後の方針

各 GPU 間のデータ依存が起こらないようなイタレーション処理の並列化から対応を行い，その後データ依存があるものに対してもコード生成ができるように改良を行いたい．

この研究の評価は，ベンチマークによる性能評価と実際にプログラムをした際のプログラムソースの行数の減少を持って行う事とする．またどれだけの数の GPU を対象にする点について，初期は 2 つとして，そこから数を増やしていく．

参考文献

[1] Takashi Shimokawabe,et al., An 80-Fold Speedup, 15.0 TFlops, Full GPU Acceleration of Non-Hydrostatic Weather Model ASUCA Production Code, Proceeding of the 2009 ACM/IEEE conference on SuperComputing 2010 PP.1-11.

[2] 小川慧, 青木尊之，山中晃徳, GPU クラスタを用いた Phase Field モデルに基づく相変態計算のスケーラビリティ, 計算工学講演会論文集, PP.145-148(2010).

[3] 大島聡史, 平澤将一，本多弘樹 OMPCUDA:GPU 向け OpenMP の実装, 情報処理学会研究報告, PP.121-126(2008).

[4] Seyong Lee and Rudolf Eigenmann, OpenMPC:Extended OpenMP Programming and Tuning for GPUs, Proceeding of the 2009 ACM/IEEE conference on SuperComputing 2010 PP.1-11.