2015 年 6 月 1 日 情報システム基盤学合同輪講発表資料

GPUdmm: A high-performance and memory-oblivious GPU architecture using dynamic memory management

著者： Youngsok Kim, Jaewon Lee, Jae-Eon Jo, and Jangwoo Kim.

出典： *High Performance Computer Architecture, 2014 IEEE 20th International Symposium on. 546-557. Feb. 2014*

発表者： 1553002 石原雅也

1 はじめに

3D グラフィックスの表示に必要な計算処理を行う際に用いる GPU を計算機の数値計算に利用する GPGPU（General-purpose computing on graphics processing units）に注目が集まっている。これは、CPU と比べ機能は限定されているが、大量のデータを高速に並列処理することができるためである。その需要に伴い、GPU の高性能化も進んでいる。一方で、GPU のメモリの管理システムは大容量化に囚われ困難なプログラムを強いるシステムになってしまっている。

本稿では、従来のメモリの管理システムに囚われない新しい動的なメモリ管理を使用して、高性能かつメモリを意識することのない新しい GPU アーキテクチャを提案する。また、従来のメモリ管理システムを用いた GPU との性能の比較評価を GPGPU-Sim というシミュレーションソフトで行う。

2 研究背景

従来の GPU のメモリの管理システムとして、以下の 2 点が挙げられる [1]。

2．1 プログラマが管理するメモリシステム

プログラマが管理する GPU メモリのモデル (図 1) では、GPU で使いたいデータは GPU メモリになくてはならない。そのため、プログラマが自分で CPU-GPU 間の転送を管理して、CPU から GPU にデータを持ってくるように計画することになる。CPU メモリから GPU メモリにデータを一度持っていき、計算を行ったのち再び GPU メモリから CPU メモリに戻す、といった流れとなる。

この方法のメリットとしては、計算に必要なデータをあらかじめ GPU メモリに持ってきているため、何回もアクセスが必要な計算の場合、データアクセスの時間が短縮出来る点が挙げられる。逆にデメリットとしては、あらかじめ GPU で実行するカーネル関数が動く前に GPU メモリにデータをもってきておかなければならず、その作業をすべてプログラマーがスケジューリングして決めるため、負担がかかってしまう点が挙げられる。そして、CPU メモリからデータを転送する際、計算に不必要なデータもまとめて送ることになってしまうため、例えば GPU メモリの容量が小さい場合、何回かに分けて転送しなければならなくなってしまい、結果時間がかかってしまう点も挙げられる。また、カーネルの実行中は CPU/GPU 間の転送が著しく遅くなってしまうため基本的に作業の重複は難しくなってしまう。

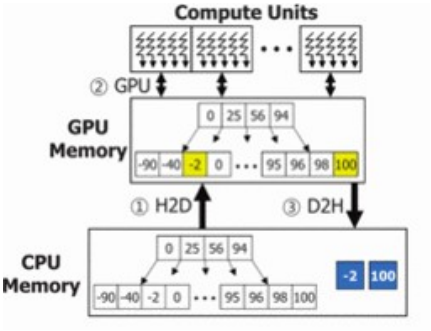

図 1：プログラマが管理するメモリシステム

2．2 ピンドホストメモリシステム

ピンドホストメモリは、GPU メモリを無効にして、GPU が CPU メモリと直接アクセスしようという方法である。GPU メモリを介することなく、直接 CPU メモリとデータアクセスを行い計算する（図 2)。

この方法のメリットとしては、前述の方法のように GPU メモリに前もってデータを用意する必要がないため、プログラマーの負担が圧倒的に軽減される。また、必要なデータだけを直接とりにいくため、不必要なデータを取る必要がなくなる。一方、すべての動作を直接 CPU メモリと行うため、同じデータに何回もアクセスが必要な計算を行う場合、データの転送時間が増えてしまう点が挙げられる。

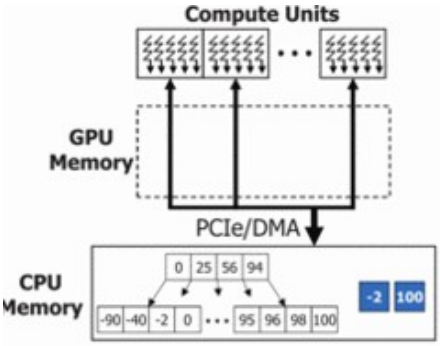

図 2：ピンドホストメモリシステム

3　新しいメモリ管理システム

従来のメモリ管理システムでは高性能化とプログラムの簡便化を図るのに限界が生じる。そのため、従来の二つのシステムのメリットを取り込んだ上で新しいメモリ管理システムの考案が必要となる。

そこで、本論文では新しいメモリ管理システムを提案している。GPU メモリを CPU メモリのキャッシュとして扱うことで、アクセス時間の短縮を行うとともに、頻繁にアクセスするデータのみを GPU メモリ上に配置して、CPU に未使用のデータを配置するのを自動でしてくれるようなメモリシステムを構築する。こうすることで、プログラムの簡便化を図る一方で、高性能化を実現できると考えた。

4　GPUdmm アーキテクチャ

ここでは、新しい GPU アーキテクチャとして GPUdmm を提案する。従来の GPU のアーキテクチャをベースラインに、新しいメモリ管理システムを可能とするためメモリパーテーションの変更を行った (図 3)。

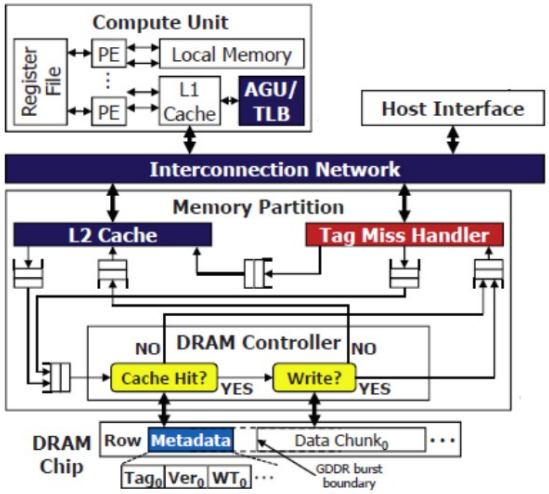

図 3：GPUdmm アーキテクチャ

L2 キャッシュからのメモリアクセスを DRAM コントローラーが受信すると、まずはその目的のデータが DRAM にあるかどうかメタデータを使って検索する。ここで、キャッシュヒットした場合、次にそのアクセスがリードかライトかを判別することになる。

リード場合、キャッシュヒットした場合は、そのまま対象のデータにアクセスを行い、リードであるため L2 キャッシュに転送される。キャッシュヒットしなかった場合は、そのままタグミスハンドラーに転送される。その後、CPU メモリからデータを GPU メモリに送信してもらったのち、タグミスハンドラーから DRAM コントローラーにデータアクセスが転送される。そして改めてキャッシュヒットを行い、L2 に転送される。

ライトの場合、キャッシュヒットした場合は対象のデータにアクセスしたのち、ライトなのでタグミスハンドラーへ転送される。その後、ライトスルーに設定されていた場合はタグミスハンドラーにて CPU メモリと L2 キャッシュの両方へ転送される。ライトバックの場合は L2 キャッシュにのみ転送される。キャッシュヒットしなかった場合も、タグミスハンドラーへ転送される。その後、タグミスハンドラーにて前にそのアクセスがあったかを確認し、すでにあった場合はそのままタグミスハンドラー内で保留とされる。初めてのアクセスの場合は、一度 CPU メモリに送信され、L2 キャッシュに転送される。

5　性能評価

GPU アーキテクチャのシミュレーションソフト、GPGPU-Sim[2] を用いて、プログラマ管理モデル、ピンドメモリモデル、GPUdmm の 3 つについての性能の比較評価を行った (図 4)。異なるベンチマークソフトを用いて状況別 (データの局所性の高低、大領域かつランダム) に評価をしたところ、ほとんどの場合について高性能化が認められた。とりわけ広大な領域でランダムアクセスするベンチマークにおいて、従来の GPU の 5 倍もの性能アップが認められた。

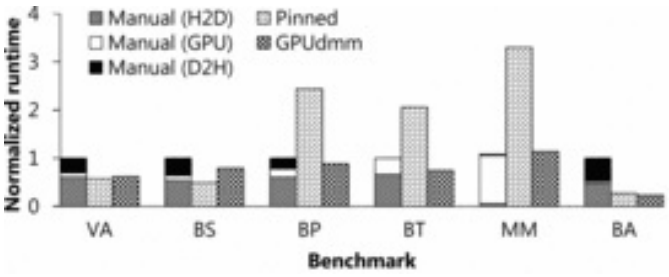

図 4：GPUdmm の比較評価結果

6　まとめ

従来のメモリ管理システムにおいてプログラムの簡便化と高性能化の両立は最も重要な課題の一つである。その課題を解決するために、今回は高パフォーマンスかつメモリを意識することのなくなった簡便なプログラミングを可能としたメモリ管理システムと GPU アーキテクチャを提案することができた。

参考文献

[1] J. Fix, A.Wilkes, and K. Skadron. Accelerating Braided B+ Tree Searches on a GPU with CUDA. In A4MMC, 2011.

[2] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt. Analyzing CUDA Workloads Using a Detailed GPU Simulator. In ISPASS, 2009