

GPGPU-Sim の高速化

川井 博斗[†] 味曾野 智礼[‡] 吉瀬 謙二[‡][†] 東京工業大学 工学部情報工学科[‡] 東京工業大学 大学院情報理工学研究所

E-mail: {kawai, misono}@arch.cs.titech.ac.jp, kise@cs.titech.ac.jp

1 はじめに

近年、GPU は画像処理の分野のみならず様々な高性能計算のために使用されており、その需要度は今後も増加していくと考えられる。GPU を含むマイクロアーキテクチャ研究においては、シミュレーションによって性能評価を行うことが一般的である。しかしながら、既存の GPU シミュレータは実機での実行に比べ数千から数万倍以上の時間を必要とするという問題があり、研究基盤は充分であるとは言えない。その高速化は解決すべき問題の1つである。GPGPU-Sim [1] は最もよく使用されている GPU シミュレータの1つである。本稿では、その高速化について検討を行う。

2 GPGPU-Sim

本章では、まず GPGPU-Sim の概要について述べ、その後、実機との実行時間差について述べる。

2.1 概要

GPGPU-Sim は機能レベルまたはサイクルレベルで動作する GPU シミュレータであり、ランタイムライブラリを置き換えることによって、CUDA や OpenCL のプログラムをそのまま実行することができる。アーキテクチャは NVIDIA の GPU、特に Fermi アーキテクチャ[2] をターゲットとしており、コアの数やキャッシュサイズ等を変更することができる。機能レベルのシミュレーションでは、実行サイクル数などの計測は行わず、GPU プログラムを実行する。一方でサイクルレベルの場合、モデルとしたアーキテクチャにおける実行サイクル数やキャッシュのヒット率などのデータを得ることができる。GPGPU-Sim のマニュアル [3] によれば、実機と比較した IPC の差は数%である。GPGPU-Sim の構成は SIMT コア、インターコネクションネットワーク(ICNT)、L2 キャッシュ、DRAM の4つの領域に分かれており、それらは異なるクロックで動作する。SIMT コアとは Fermi アーキテクチャのストリーミング・マルチプロセッサ (SM) に相当するものであり、この部分が計算を実行する。GPGPU-Sim のメインループでは、そのシミュレーションサイクルで動作する領域のシミュレーションが行われる。

各 SIMT コアは図 1 に示すように fetch ステージから writeback ステージまで 6 段のパイプラインになっている。シミュレーションを行う際は、writeback ステージからパイプラインの構成とは逆の順番に実行する。なお、GPGPU-Sim では execute ステージではなく issue ステージにて対象の命令を実行し、シミュレーションを行っている。ただし、ロード・ストア命令については execute ステージにおいてメモリのアクセスを行っている。

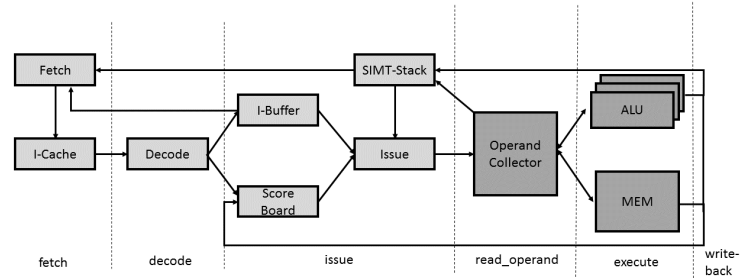


図 1: SIMT コアのパイプライン

表 1: 実機とシミュレーションの実行時間差

アプリケーション	実機 (ms)	GPGPU-Sim(ms)	時間差
b+tree	6.4	267702	41821
backprop	258.5	303984	1175
dwt2d	375.9	370151	984
kmeans	473.3	17233779	39400
nw	298.7	914776	3061

る。他の命令は execute ステージでは、実行に必要なとなるサイクル数をカウントしているだけである。

2.2 実機とシミュレータの実行時間差

GPGPU-Sim と実機の GPU において Rodinia ベンチマーク [4] のサブセットを実行し、シミュレーション時間を測定した。その結果を表 1 に示す。評価環境は CPU は Intel Core i7 3770K(4-core), メモリが 16GB, OS は Ubuntu 14.04, 64bit である。また、実機の GPU は NVIDIA GeForce GTX660, CUDA7.0 を、GPGPU-Sim では NVIDIA GeForce GTX480 のモデルで CUDA4.2 を使用している。結果より、GPGPU-Sim は実機に対して、数千倍から数万倍の実行時間を必要とすることがわかる。

このように大きく実行時間差が出る理由は、GPU と CPU の構成の違いのためである。GPU は単純な計算ユニットの数が CPU よりも遥かに多い。特にサイクルレベルのシミュレーションを行う場合、非常に多くのハードウェアコンポーネントのシミュレーションを逐次的に CPU で行う必要があるため多くの実行時間が必要となる。

Acceleration of GPGPU-Sim

Hiroto KAWAI[†], Tomohiro MISONO[‡], and Kenji KISE[‡][†]Department of Computer Science, Tokyo Institute of Technology [‡]Graduate School of Information Science and Engineering, Tokyo Institute of Technology

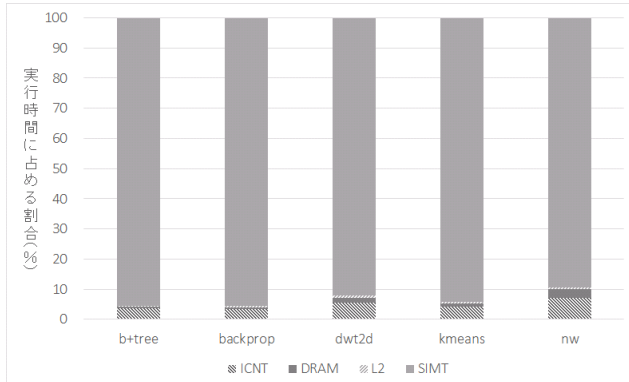


図 2: クロック別領域の実行時間使用率

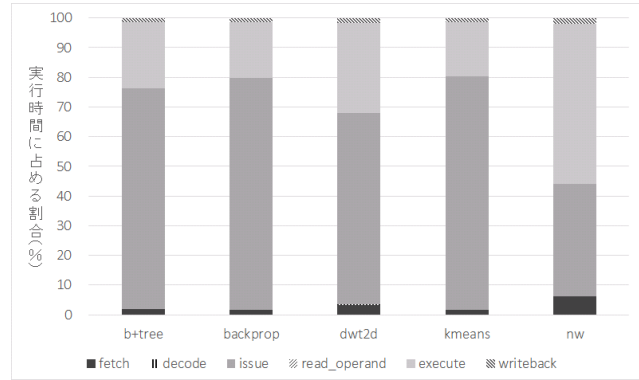


図 3: パイプラインステージの実行時間使用率

3 高速化箇所の検討

本章では、GPGPU-Sim の高速化を検討するため、シミュレーションの実行を Intel Vtune Amplifier 2015[5] を用いてプロファイリングし、各関数のシミュレーション時間の割合を測定した結果について述べる。測定した点は以下の2つである。

- ICNT/DRAM/L2 キャッシュ/SIMT コアのクロック別領域
- SIMT コア領域の各パイプラインステージ

3.1 クロック別領域のプロファイル

2.2 節と同じ Rodinia ベンチマークに対してプロファイルした結果を図2に示す。図より、SIMT コアがシミュレーション実行結果の大部分を占めていることが分かる。これはハードウェア規模としても SIMT コアの占める割合は ICNT/DRAM/L2 よりもはるかに大きいからためである。

この結果より、シミュレーションの全体を高速化するためには SIMT コアの計算部分を高速化することが必要であると考えられる。

3.2 SIMT コア部のプロファイル

SIMT コアはパイプライン化が行われているため、その各ステージの実行時間を測定した。その結果を図3に示す。図より、nw のベンチマーク以外では issue ステージが最も割合が高いことが分かる。これは、issue ステージにおいて GPGPU-Sim がプログラムの実際の命令を実行するためである。

nw のベンチマークが他のアプリケーションと異なる性質を示す理由は、他のアプリケーションに比べてロード・ストア命令が多いからであると考えられる。SIMT コアがロード・ストア命令を処理する場合、内部のレジスタがバンクとアービターを用いて管理されている。複数のユニットが同時にレジスタにアクセスする場合、一度にデータのやり取りをすることができない。そのため、各サイクルごとにレジスタアクセスが衝突しないか確認を行う必要があり、その過程を execute ステージにおいて行っている。頻繁にロード・ストア命令を実行する場合はこの確認回数が増え、より多くの時間が必要となり、execute ステージの実行に最も時間を必要としたと考えられる。

以上より、一般的には issue ステージの高速化を行うことが最も効果的であると考えられる。しかしながら、ロード・ストア命令の多いアプリケーションにおいては、issue ステージだけでは、あまり性能向上が期待できない。issue ステージと execute ステージを合わせた割合はいずれのアプリケーションでも 90 % 以上の高い割合を占めるため、両方のステージを高速化することが必要であると考えられる。

4 まとめ

本稿では、GPGPU-Sim でベンチマークを実行した際のプロファイル結果からその高速化箇所について検討を行った。

今後の予定は、異なるベンチマークを実行した際の実行時間の比較を行うこと、それらプロファイルの結果を考慮したうえで高速化のための変更を行い、評価をとることが考えられる。

参考文献

- [1] A. Bakhoda, G.L. Yuan, W.W.L. Fung, H. Wong, and T.M. Aamodt. Analyzing cuda workloads using a detailed gpu simulator. In *IEEE International Symposium on Performance Analysis of Systems and Software, 2009. ISPASS 2009.*, pp. 163–174, April 2009.
- [2] Nvidia Fermi Architecture. http://nvidia.com/content/pdf/fermi_white_papers/p.glaskowsky-nvidia's_fermi-the_first_complete_gpu_architecture.pdf/.
- [3] GPGPU-Sim. <http://www.gpgpu-sim.org/>.
- [4] Shuai Che, M. Boyer, Jiayuan Meng, D. Tarjan, J.W. Sheaffer, Sang-Ha Lee, and K. Skadron. Rodinia: A benchmark suite for heterogeneous computing. In *IEEE International Symposium on Workload Characterization, 2009. IISWC 2009.*, pp. 44–54, Oct 2009.
- [5] Intel Vtune Amplifier. <https://software.intel.com/en-us/intel-vtune-amplifier-xe/>.