コード移動に基づく分岐発散低減

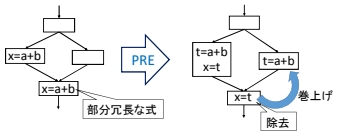
福原淳司　滝本宗宏
東京理科大学大学院　理工学研究科　情報科学専攻

背景

- GPUプログラミングの普及
- GPUの多くはSIMD型の実行形式

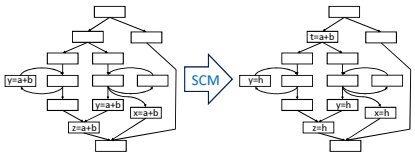
GPUの実行効率を低下させる分岐発散が生じる可能性がある

部分冗長除去(PRE)



Sparse Code Motion(SCM)

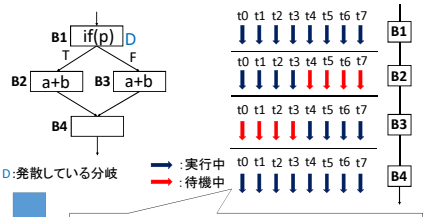
Lazy Code Motionに基づく部分冗長除去を行った上で、計算式の疎なコード配置を実現する。
計算式が現れる回数を最小限に抑えるように変形する。



分岐発散

SIMD処理で同一ウォープ内のスレッドが異なる分岐先に分岐するときに生じる実行効率の低下のこと。
分岐発散が発生すると、各分岐先の命令を逐次実行する。
一方の経路に分岐したスレッドが命令を実行している間、もう一方の分岐経路に従うスレッドは休止状態となっている。

並列性が失われ、GPU使用率が低下



分岐条件pがtrueであるスレッドt0～t3がB2の命令を並列実行し、その間分岐条件pがfalseであるスレッドt4～t7は休止している。次に、スレッドt4～t7がB3の命令を並列実行し、その間スレッドt0～t3は休止している。B2を実行したあとにB3を実行するという逐次実行になっている。

分岐発散が発生しているプログラムに対しては、特定の経路にコードを挿入する従来の部分冗長除去法は分岐発散による実行効率の低下を増大させる場合があり、適用が難しい。

目的

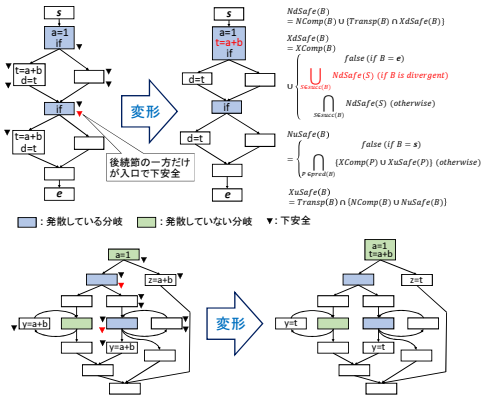
分岐発散によるGPUの実行効率の低下を低減し、実行速度の改善を図る。

提案手法

- 計算式の疎なコード配置を実現するSCMと、投機的なコード移動を組み合わせる
- 分岐発散していない箇所については、従来のPREの振る舞いをする
- 分岐発散しているブロックの出口では、後続節のいずれか一方の入口で下安全であれば、下安全になるようにデータフロー方程式を変更

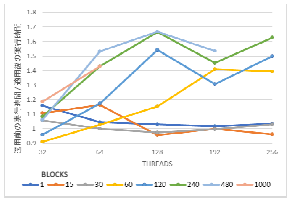
各分岐経路にある計算式の巻上げを行うことができ、分岐発散によって分岐の外に出してもコストが増えない式を投機的に移動させることによって、挿入によらず除去できる式の数を増やす

PREによって多くの冗長な式を除去するだけでなく、分岐発散による実行効率の低下を低減する



実験

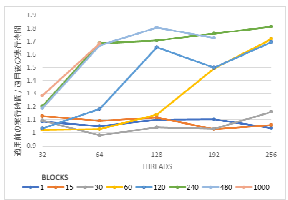
本手法をMCML(Monte Carlo Modeling of Light Transport in Multi-Layered Tissues)に適用し、適用前後で実行速度を計測した。
最大で約1.8倍の実行速度の改善が得られ、本手法の有効性を確認した。



実験結果1: nvcc -O2との実行速度比較

実験環境
- OS: Ubuntu 16.04 LTS
- CPU: Intel Core i7-4770K
- GPU: Geforce GTX TITAN Black
- CUDA Toolkit 5.0
- g++ 5.4
- Ocelot version 2.1.0

適用前: nvcc -O2+分岐融合
適用後: nvcc -O2+分岐融合 +本手法



実験結果2: nvcc -O2 + Global Value Numbering (GVN)との実行速度比較

実験環境
- OS: Ubuntu 16.04 LTS
- CPU: Intel Core i7-4770K
- GPU: Geforce GTX TITAN Black
- CUDA Toolkit 5.0
- g++ 5.4
- Ocelot version 2.1.0

適用前: nvcc -O2+分岐融合 +GVN
適用後: nvcc -O2+分岐融合 +GVN+本手法